EURECOM
Networking and Security department
Campus SophiaTech
CS 50193
06904 Sophia Antipolis cedex
FRANCE

Research Report RR-15-305

Taming the Android AppStore: Lightweight Characterization of Android Applications

April 27th, 2015

Luigi Vigneri[†], Jaideep Chandrashekar[‡], Ioannis Pefkianakis[‡] and Olivier Heen[‡]
[†]EURECOM, [‡]Technicolor Research

Tel : (+33) 4 93 00 81 00
Fax : (+33) 4 93 00 82 00

[1]EURECOM's research is partially supported by its industrial members: BMW Group Research and Technology, IABG, Monaco Telecom, Orange, Principaut de Monaco, SAP, SFR, ST Microelectronics, Symantec.

Taming the Android AppStore: Lightweight Characterization of Android Applications

Luigi Vigneri, Jaideep Chandrashekar, Ioannis Pefkianakis and Olivier Heen

Abstract

There are over 1.2 million applications on the Google Play store today with a large number of competing applications for any given use or function. This creates challenges for users in selecting the right application. Moreover, some of the applications being of dubious origin, there are no mechanisms for users to understand who the applications are talking to, and to what extent. In our work, we first develop a lightweight characterization methodology that can automatically extract descriptions of application network behavior, and apply this to a large selection of applications from the Google App Store. We find several instances of overly aggressive communication with tracking websites, of excessive communication with ad related sites, and of communication with sites previously associated with malware activity. Our results underscore the need for a tool to provide users more visibility into the communication of apps installed on their mobile devices. To this end, we develop an Android application to do just this; our application monitors outgoing traffic, associates it with particular applications, and then identifies destinations in particular categories that we believe suspicious or else important to reveal to the end-user.

Index Terms

Computer science; Networking; Android; Privacy; Traffic monitoring; Tracking systems; URLs analysis.

Contents

List of Figures

1 Introduction

The two dominant mobile application ecosystems today, Apple iOS and Android, reflect very contrasting philosophies. In the former, applications are vetted against a defined set of acceptable use behaviors before being released on the store. In contrast, the Android based app stores advocate a more libertarian approach and apply a much looser set of guidelines (mainly focusing on keeping out malicious applications). The Google Play Store, the largest and most prevalent Android marketplace today, contains over 1.2 million distinct applications, a number of which are of dubious nature (even if not malicious). The average Android phone user faces a daunting set of questions while installing an application. *Which of the many similarly named apps should I install? Does it report information to online trackers? Does it have too many ads?* and so on. A search for "weather" apps returned well over 30 applications all of which contained `weather` in the name; a large number of these are rated with 4 or more stars (out of 5). Having installed the application, the user has no visibility into who the application is actually communicating with, and whether this complies with the app's intended purpose.

Our goal is to build a system to characterize the network behavior of Android applications. This characterization could inform users about how the application is expected to behave when installed – useful information when selecting an application. Given our focus on network behavior, we are interested in identifying the *kinds* of destinations connected to, whether the application connects to a large number of ad sites, how often it talks to online tracking sites, and whether it communicates with sites that have been deemed suspicious.

In this paper, we first develop a methodology that enables us to characterize a large number of applications quickly. In gaining scale, we necessarily sacrifice a small level of accuracy. Applications connecting to suspicious websites may not actually be sending any private data over the connection. While methods based on taint tracking and static program analysis [2, 4, 8, 12, 14, 15, 17] enable more accurate characterization of actual data exfiltration, they are intrusive and hard to scale. We view our own work as complementary to such methods, providing a first level characterization that can enable applications to be selected for further, more detailed inspection. We focus on 3 distinct characteristics, which we believe to be undesirable to end-users, of the destinations being contacted: (i) if they are ad-related, (ii) if they relate to tracking users, or (iii) if the domains have previously been associated with malware or other suspicious activity.

We then apply this methodology to characterize a large sample of *free* applications from the Google Play Store, across different application categories. Our results uncover a great deal of diversity in behavior: some applications connect to almost 2000 different URLs in a few minutes of execution while others generate almost no network traffic. Further, we also identify applications that involve an extensive level of tracking, and those that make an inordinate number of connections to ad related sites. We also identify application instances that make connections to websites that have previously been associated with malware activity. These results

underscore a crucial shortcoming, and this is a lack of effective tools and mechanisms to audit installed applications and to provide users' greater visibility into application behavior. To this end, we develop a monitoring application (NSA), that identifies particular types of destinations being connected to by installed applications. We have made a version of the application available to reviewers through an anonymous URL.

The rest of the paper is as follows: In Section 2, we provide some background on the types of domains that we are interested in characterizing. In Section 3, we describe related work in the area and put our own work in perspective. In Section 4, we elaborate on the process used to collect our dataset and provide some high level summaries. In Sections 5 and 6, we present a detailed analysis of the dataset. Section 7 contains a high level description of the architecture and operation of the monitoring application we developed, and we conclude in Section 8.

2 Background

This work focuses on the network behavior of Android applications. To this end, we are interested in the types of destinations connected to by the application. Our methodology, at a high level, consists of extracting network traces from short executions of mobile applications, extracting URL endpoints from the network traces, and categorizing these URLs. To this end, we focus on three distinct types of URL categories which we describe below:

Advertising related sites: Much of todays online economy is driven by advertising revenue. Web publishers, and mobile app developers, auction space on their websites, or screens, and the ads are delivered by so-called ad networks. Particularly with (*free*) mobile applications, advertisements can support app development and support costs, and is a very popular model in the Google Play marketplace. However, most users view advertisements as intrusive and associate negative connotations to them [18]. In order to categorize a URL as being ad-related, we rely on the `EasyList`[1] set of filters that are available from AdBlock [1]. This set is able to identify the vast majority of ad serving URLs and can also identify – based on the URL pattern – banner ads and advertisements delivered by other means (javascript, frames, etc.).

Tracking sites: These provide a mechanism for online web services and third parties to "follow" users over multiple sessions, and over different websites. While websites have traditionally used cookies, the app ecosystem uses more direct forms of identification such as UDID or other device identifiers which are made available through the OS APIs. The issue of online tracking has been vigorously debated in the recent past, and privacy advocates argue that it allows for open ended profiling of end-users. Importantly, users are rarely aware of the actual entities that are

[1] `http://easylist.adblockplus.org`

tracking them, and to what degree and the tracking ecosystem today lacks transparency.

To identify such URL endpoint, we rely on the `EasyPrivacy` set of filters, which are also available through Adblock as an optional subscription. This set covers a large variety of tracking mechanisms (web bugs, tracking code, beacons, etc.).

Suspicious sites: We use this third category as a catch-all term for any sites that are associated with any form of malware or illicit content, and we do this for the following rationale. Typically, a mobile application is likely to connect to different URLs to carry out the functionality related to the application, apart from the URLs related to ads and tracking. If the application is benign and completely legitimate, it is likely to connect only to destinations that are trusted and safe. However, if the application does make connections to particular websites that, through other channels, have been deemed suspicious or malicious, it is unlikely that the application is completely benign. This likelihood grows larger as the connections to such destinations increases. While not necessarily very accurate – false positives can happen – we believe that this factor is one of many that must be considered while vetting the application. Similarly, the mere fact of a connection to a "malicious" website may not be evidence of private information being passed on; however, it does arouse suspicion.

In our work, we rely extensively on the VirusTotal meta classification engine [20], which acts as a front end for a large number of AntiVirus, Spam & Phishing blacklist, and Malware analysis engines. For each URL identified in the pcap trace, we issue a query to VirusTotal and obtain two types of information for the URL: (i) an aggregate response from all the back-end engines indicating the suspiciousness level of the URL domain, and (ii) a categorization of the URL domain into one of a set of 68 categories. Both of these are obtained from the query responses returned by VirusTotal.

3 Related Work

Previous work in mobile application monitoring falls into three broad areas, which we briefly discuss and further contrast them with our work.

Application profiling: Given the lack of insights associated with the Android app store, a number of studies have focused on profiling mobile apps. In [21], authors describe a multi-layer profiling approach that covers both system and network aspects. While capable of obtaining detailed behavioral profiles, the methodology is difficult to scale to a large number of applications, and cannot be implemented as an Android app. In contrast, our (lightweight) approach can characterize a large number of applications. Different from [21] our study focuses on many aspects of the destinations being connected to by the app. In [6] the authors describe techniques to fingerprint mobile apps based on their network behavior. Our work does

not attempt to find application signatures, but to characterize and compare the network behavior of different apps.

Privacy/security auditing: Several studies have looked at the issue of identifying privacy leaks in mobile applications. TaintDroid [9] applies taint tracking mechanisms inside the Android Dalvik VM to identify instruction sequences where a particular input leaves the system. The work revealed that roughly half of the tested apps reported the user's location to advertising servers. Apart from TaintDroid, there are several systems which seek to identify privacy leaks in iOS [8] and Android smartphones [2, 4, 12, 14, 15, 17], using the taint analysis approach. While comprehensive, the above approaches often require significant changes in mobile device OS, or phones to be rooted, which limits their applicability. Yet another shortcoming with these approaches is that they potentially miss communication to suspicious third party destinations (e.g., trackers) when they do not leak traffic.

Different from taint analysis approaches, SpanDex [5] extends the Android Dalvik Virtual Machine to ensure that apps do not leak users' passwords. SpanDex analyzes implicit flows using techniques from symbolic execution to quantify the amount of information a process control flow reveals about a secret. ipShield [3] performs monitoring of every sensor accessed by an app, and uses this information to perform privacy risk assessment. Both SpanDex and ipShield require modifications in the mobile device OS, while they are not focusing on suspicious destinations. Finally, in [11] the authors study the privacy and security risks posed by embedded or in-app advertisement libraries, used in current smartphones. Our study is much more generic, seeking to identify various characteristics of the mobile apps network behavior.

Traffic characterization: Existing work has been focused on analysing smartphone application usage (e.g., geographic coverage, mobility, traffic volume vs. app category) [22] and traffic characteristics [10, 16] of mobile devices. None of these studies seek to characterize app communication with third party websites.

4 Dataset

At a high level, our methodology involves selecting, downloading and executing an application on an *unrooted* Android phone, and capturing all the network activity during its execution. The network activity is post-processed to extract all the URLs contacted by the application, which are then categorized and classified using a number of existing online engines. In the rest of this section, we describe the methodology in detail.

4.1 Application Selection

The (roughly) 1.2 million applications in the Google App Store today span 25 different categories. The choice of category for an application is left to the app developer. As of July 2014, the available categories in the app store are enumerated

Game	News_and_magazines	Comics
Libraries_and_demo	Communication	Entertainment
Education	Finance	Lifestyle
Books_and_reference	Medical	Weather
Media_and_video	Music_and_audio	Tools
Personalization	Photography	Productivity
Business	Health_and_fitness	Shopping
Social	Sports	
Transportation	Travel_and_local	

Table 1: App Store Categories (July 2014)

in Table 1. In each of the categories, the applications can be listed in various orderings – most popular, most highly rated, newest, etc., and these are accessible with a number of third party APIs. For our characterization, we selected the top 100 *most popular* applications and the top 100 *newest* applications in each of the categories available, i.e. a total of 5000 applications. However applications can belong to both *newest* and *most popular* sets, so the actual number of downloaded applications is smaller. While not exhaustive, our application set represents a reasonably good sample of the app store (and its categories).

4.2 Application Execution

From the set of applications selected and downloaded, we filter out all the applications that do not have the `INTERNET` permission property set in the manifest since these applications would not be able to generate any traffic. Each of the remaining applications is downloaded and executed on a Samsung Galaxy SIII Mini GT-I8190 smartphone running Android version 4.1.2, which was configured with a VPN client (*OpenVPN for Android*) connected to an external VPN server. All traffic generated on the smartphone transits through the VPN server, where it is captured using `tcpdump`. The manner of network traffic capture differentiates our work from previous work in this field which captured traffic *locally* on the smartphone; this is inherently restrictive. The smartphone is connected to a PC, and the app is launched using the `adb` tool on the command line. In order to simulate user interaction with the launched and running application, we use `monkey`, a command-line tool, to generate a series of 10000 user interaction events (screen touches, scroll actions) in two phases, with a short gap of 50 seconds between. This takes care of the situation that some applications have a start-up delay before actuation. A `tcpdump` process is coordinated on the VPN server with each app launch, and thus we obtain a pcap file for each app execution. It is important to point out that while unlikely, there may be other background traffic that co-occurs with application generated traffic. We exercised due diligence in removing all non-essential applications from the smartphone. In addition, we recorded traffic for a 24 hour period without any apps installed on the phone and recorded all traffic generated. We filter out any URLs observed in this trace from each application trace, if found.

4.3 URL Analysis

We process each packet capture with `tshark` to extract the complete set of HTTP URLs in the trace. Each of these URLs is classified along three different dimensions, following the previous section (§2). Note that we filter out HTTPS traffic (which has been shown to be small in Android apps [21]) and only consider HTTP traffic; HTTPS traffic does not expose the headers that we analyze. For each URL extracted, we carry out three different checks as follows:

1. We check the URL against the set of descriptors in `EasyList` and, if a match is found, we classify the URL as being ad-related, since the connection most likely was made to retrieve an ad-element to display on the smartphone screen.

2. We check the URL against the set of filters contained in `EasyPrivacy` and, if a match is found, we mark the URL as being *tracking related*.

3. Finally, we issue a query to the VirusTotal service with the URL as a parameter to obtain a reply that aggregates the findings of all of the backend engines supported by VirusTotal. In addition, we also extract fully qualified domain names from the URL and query VirusTotal for information about these. The results relevant to domain names include things such as the `Webutation` safety score for a domain, and so on.

Finally, our dataset consists of 2146 processed applications (1710 with traffic activity), spanning 25 distinct application categories and which in the aggregate, connect to almost 250k unique URLs and across 1985 top level domains.

5 Application Destination Characterization

While there is a considerable body of work in the area of profiling mobile apps, the focus has been on detecting data leakage, or on developing behavioral fingerprints of the applications. There has been relatively less work on characterizing the applications in terms of the network destinations they visit, and the nature of these destinations. We focus on analyzing the network end-points in depth and on understanding similarities (or differences) in certain app categories in terms of this behavior. We start by presenting some high level statistics of network end-points, across the set of applications analyzed.

Apps URLs and domains: We see a tremendous range in application behavior: a large number of applications generate no traffic at all while some applications generate well in excess of 1000 HTTP requests. We find the app `Music Volume EQ` connects to almost *2000 distinct URLs*. Interestingly, Music Volume EQ is a volume slider app, and not an app that would really require access to the network. By all accounts, these numbers are large especially considering that our methodology does not support authenticating against user accounts (such applications will not progress beyond the login screen). Fig. 1(a) shows a distribution of the number of URLs visited by each executed application. From the figure, about 10% of the apps tested connect to more than 500 *distinct* URLs (recall that the execution of

Application Name	URLs
Music Volume EQ	1958
signal.booster.conchi...	1882
Entranements Quot. FREE	1827
simulateur laser	1657
The Weather Channel	1544
France TV Replay	1465
Gestion du budget	1415
Morandini Blog	1403
FR24 Premium	1350
cart.tabs.sw	1275

Application Name	TLDs
Morandini Blog	113
com.issakol12i.myapp	104
Prof Orientation Test	103
Exercice Machines Gym	101
Motor Racing News	93
Le Tlgramme - Actualit	92
app20192.vinebre	82
Music Explorer	70
PowerAMP Music Player	69
Entrainements Quot. FREE	63

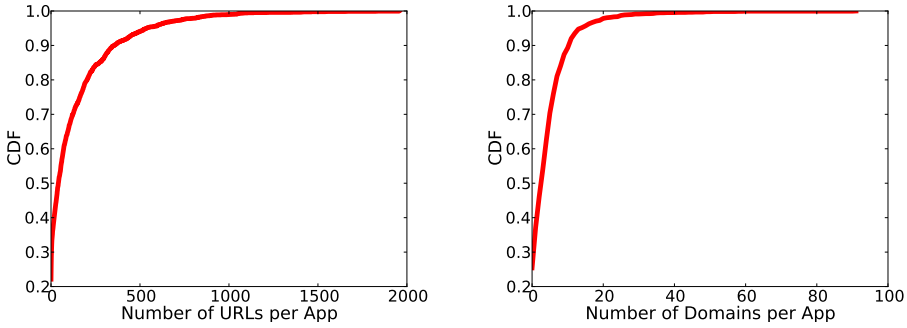
Table 2: Top 10 Applications, by URLs (left) and by Top Level Domains (right)

each application only lasts a few minutes). This level of "chattiness" significantly impacts resource usage on the mobile device. Interestingly, we still identify apps which do not engage in network activity, although they declare (in manifest file) that they require network access.

In Table 2, we enumerate the top 10 applications seen in our dataset, ranked by the number of URLs connected to (at least 25 applications connect to more than 1000 URLs during execution). These applications are very diverse, from weather to music and budget. This confirms the need to consider broad and varied dataset rather than focusing on specific categories.

Multiple URLs can correspond to the same domain. The number of distinct domains, apps connect to, captures the different activities carried out inside the application. Across the applications in our dataset, the median number of domains connected to is 4, while some apps connect to more than 100. For example, `Morandini Blog`, which is a blog reader application, communicates with 113 distinct domains. Interestingly, it connects with 6 *different* ad networks, along with a number of analytics and tracking websites. In figure 1(b), we look across applications and plot the distribution of domains communicated with by each application. About half of the apps connect to 4 or fewer domains, and we also see significant variability across applications. Roughly 10% of the apps connect to 20 or more domains over the execution window. Table 2 enumerates the top 10 apps ranked by the number of distinct domains connected to. Rows marked in italics denote apps that also happen to fall in the top 10 when ranked by the number of URLs communicated with (cf. Table 2).

Looking across applications, Table 3 enumerates the 20 most frequently contacted domains, which provides some insights about the nature of communication between the application and website. Unsurprisingly, 9 of the top 10 in this set correspond to various web services run by Google. The most popular domain in the list, `doubleclick.net`, is an advertising platform that tracks end-users, and also serves up advertisments. While `Google.com` is generally considered as the search engine portal, in our traces we found two predominant patterns associated with this particular domain: (i) `www.google.com/images/cleardot.gif?zx=<str>`, which correspond to 1x1 tracking pixels, and (ii)



(a) Number of URLs per app. (b) Number of domains per app.

Figure 1: URL and domain counts by application

doubleclick.net	0.415	ajax.googleapis.com	0.058
google.com	0.358	flurry.com	0.056
gstatic.com	0.354	nend.net	0.051
admob.com	0.266	xiti.com	0.048
googlesyndication.com	0.238	facebook.com	0.048
google-analytics.com	0.172	youtube.com	0.041
ggpht.com	0.170	scorecardresearch.com	0.041
fonts.googleapis.com	0.139	inmobi.com	0.037
googleusercontent.com	0.128	ytimg.com	0.034
samsungvideohub.com	0.088	twitter.com	0.033

Table 3: Top 20 popular domains (with fraction of applications connecting to them)

`www.google.com/ads/user-lists/<id>/?script=<num>&random=<num>`, which seems to indicate some form of user tracking.

While enumerating the communicating domains can be quite instructive, it does not reveal much about the nature of the communication between app and domain. Understanding the *type* of domain (or category) of the domain can yield a better sense of this communication. Typically, web domains are set up for well defined functions (e.g., `doubleclick.net` as an ad platform, `google-analytics` as a tracking and analytics service, etc.) and communication between the app and domain is generally consistent with the service offered by the domain. We use the following methodology to identify domain categories. First, we classify each URL as a *tracker URL*, *ad related* or *other*. In the first two cases, the services are obvious, and we consider them independent categories. In the latter case, *other*, we extract the fully qualified domain name from the URL and rely on the service provided by `Websense.com` to obtain a characterization (i.e., category) for the domain in question. Specifically, we first examine each URL and extract the fully-qualified domain name (FQDN) embedded in the URL. Then we gather all the Websense categories corresponding to each of the FQDNS that correspond to the same top level domain, and assign the majority class as the category for the top level domain. Going back to the domains listed in Table 3, we find that the most

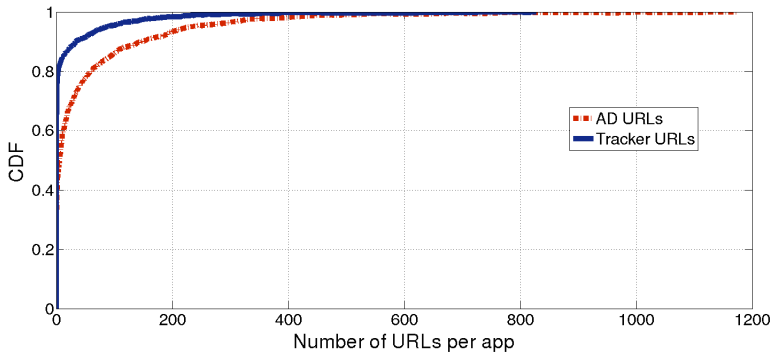


Figure 2: Ad URLs and tracker distributions

Top Level Domain	Popularity
doubleclick.net	37,153 (50.6%)
gstatic.com	16.532 (22.5%)
admob.com	3603 (4.9%)
smartadserver.com	3411 (4.6%)
inmobi.com	1399 (1.9%)

Table 4: Top 5 ad related Top Level Domains

of the 20 domains correspond to the category advertisements. In the rest of this section, we examine various types of destinations based on their domain categorization.

Ad related sites: Recall that we use AdBlock's EasyList subscription to identify advertising related destinations. Figure 2 shows the distribution of the number of ad URLs visited per app. We observe that 33% of the apps do not communicate with *any* ad destinations. On the other hand, we also see apps that connect to a very large number (>1000) of ad URLs. Overall, the average number of ad URLs associated with an application is about 40. Examining the domains of ad URLs, we find that the three most prominent ad related domains are all part of Google, as listed in Table 4. Thus, while Google does not directly make any revenue from Android itself (which is openly licensed to manufacturers), it is able to extract revenue from the ads business around the ecosystem. We further investigate ad URLs for particular apps, in Section 6.

User tracking related sites: We now look closely at the URLs in our dataset that correspond to destinations that track end-users and devices, as encoded in AdBlock's EasyPrivacy subscription lists. Previous studies have reported that such practices have largely negative connotations with users [13, 19]. Given this, it is rather surprising that tracking is so widespread, and more important, completely opaque to end-users. While the Do Not Track [7] policy has been proposed by consumer advocates and has gained some acceptance, the mechanism is restricted

Top Level Domain	Popularity
google-analytics.com	11,247 (44.4%)
xiti.com	8174 (32.3%)
scorecardresearch.com	1046 (4.1%)
estat.com	736 (2.9%)
bluekai.com	500 (2.0%)

Table 5: Top 5 tracking related Top Level Domains

Domain category	Popularity
information technology	453 (22.82%)
uncategorized	390 (19.65%)
dynamic content	171 (8.61%)
advertisements	164 (8.26%)
business and economy	110 (5.54%)
news and media	67 (3.38%)
shopping	61 (3.07%)
travel	48 (2.42%)
entertainment	41 (2.07%)
streaming media	41 (2.07%)
games	40 (2.02%)
sports	37 (1.86%)
search engines and portals	33 (1.66%)
reference materials	23 (1.16%)
internet radio and tv	23 (1.16%)
application and software download	20 (1.01%)
blogs and personal sites	17 (0.86%)
vehicles	17 (0.86%)
social networking	16 (0.81%)
personal network storage and backup	15 (0.76%)

Table 6: Popularity of Domain Categories

to web browsers, and does not extend to mobile apps in general.

In figure 2, we plot the distribution of tracking URLs associated with each application. We observe that while the vast majority (73.2%) of apps do not involve any communication with trackers, a small number of apps do indeed communicate with them. The number of tracker URLs per app can be more than 800. In Table 5, we enumerate the most popular domains associated with trackers, where popularity is defined as the number of tracker URLs, seen across all the apps, associated with a specific domain. In contrast to the results about ad-related destinations, we find the mobile tracking ecosystem to be significantly more fragmented, with many more players, even if the dominant player is associated with Google. We further investigate tracker URLs for specific apps, in Section 6.

Other web categories: We now examine the aggregate set of URLs after having removed those that correspond to the previous two categories. Table 6 enumerates the 20 domain categories with the highest number of domains associated with them. The most popular category, which covers about 22% of the total domains observed, is denoted *Information Technology* and this appears to cover a number

Domain category	Fraction of malicious domains
sex	33.33%
personals and dating	20%
ads	12.8%
business and economy	6.6%
reference materials	4.35%

Table 7: Malicious domains based on Webutation engine.

of miscellaneous web services. The next two identifiable categories correspond to *dynamic content* and *advertisements*, and both of these are very likely related to the online advertising ecosystem. Note that we see a large count for these even though we filter out those described in `EasyList` previously; the remaining URLs not filtered are likely due to new patterns not in *EasyList* or perhaps connections to ad related websites that do not involve ad placement inside the mobile application. Apart from these, we see small domain counts across a varied set of categories. In the next section, we examine these domain categories in detail and relate them to the category of the app itself.

URL *badness:* Finally, we explore an additional characteristic of the domains being connected to – "badness". Recall that VirusTotal aggregates results from a number of engines; these relate to the "suspiciousness" of a URL. While this term is somewhat ambiguous, the qualitative results can be explained thus: the engines used by VirusTotal independently crawl the URLs and catalog the various objects on them. URLs that host executable content that is deemed malware-like, are deemed suspicious. Note that reliably determining malicious intent is extremely challenging and quite outside the scope of our work. For our purpose, we simply quantify whether any engine marked the URL as such, and analyze this across the set of domain categories. By *suspicion score* for a URL, we denote the fraction of antivirus engines (VirusTotal uses 52 in all) that deem the URL suspicious (or malicious). Our result show 94.4% of the URLs have a (suspicion) score of 0. In the worst case, a URL was deemed suspicious by 3 (of 52) engines.

Suspicious domains: For classification of the suspicious domains, we use Webutation engine. Webutation is an open community about Website Reputation. It tests websites against spyware, spam and scams. Apart from collecting user feedback, Webutation queries various trusted engines like Google Safebrowsing or Norton Antivirus to check for malicious software and other dangerous elements. Overall, our analysis shows that *a small portion of the domains have been classified by Webutation as suspicious or malicious*. Specifically, we observe 2.5% suspicious, 2.9% malicious, 61% unsure (not a clear verdict) and 33.6% safe domains. Table 7 further shows the domain categories with the highest fraction of malicious domains. We observe that the top-3 malicious domain categories are "sex", "personals and dating" and "ads". In the following section, we devise a suspicion metric for mobile apps, and we investigate the most suspicious apps.

Application	Ads	Rate	Downloads
cart.tabs.sw	1174	-	-
VidTrim - Video Trimmer	1065	4.2	10.000.000
Simulateur laser	1019	2.1	5.000.000
Music Volume EQ	999	4.2	10.000.000
signal.booster.conchi.amplificador	940	-	-
com.HillieMelani.VideoEditor	720	-	-
Football365	700	3.9	10.000
Decibel (Sonometre reactif)	671	4.8	1.000
Nail Art Tutorials 2014	630	3.7	100.000
Veilleuse en Couleurs	538	3.8	10.000

Table 8: Top 10 apps connecting to ad URLs

6 Detailed Apps Characterization

In this section, we focus on individual applications and obtain an understanding of their behavior along the three axes discussed previously. Users' value (or are annoyed by) different things – some user's value privacy (tend to avoid applications with significant tracking), other's value security (and wish to avoid applications with suspicious or unreasonable behavior). To this end, we study the most prolific applications along these axes and gain some insight into their behavior.

6.1 Advertising Intensity

The Internet ecosystem, along with the mobile app marketplace, is largely driven by advertising revenue. The vast majority of mobile apps offer their services to the user for free, and are *directly* monetized by selling "real estate" (smartphone screen or website) on which ads are inserted. All of the applications in our dataset are "free" and we expect that the majority of them *will* connect to ad sites. This is confirmed in Fig.2, where more than 66% of the applications contact ad URLs. Some of the advertising APIs and engines are very aggressive in downloading ads into the mobile app screen. For example `AirPush` is one such infamous service and is so aggressive that the PlayStore lists a number of applications whose sole function is to detect this API and notify the user. Several mobile ad APIs collect detailed device information (OS version, IMEI, location, IP address, etc.), sometimes unknown by users. In general, end users find that ads (esp. display ads which are not targeted based on user interests) degrade the user experience of mobile apps and services.

Table 8 lists the top 10 apps ordered by the number of ad related URLs connected to. All the apps were executed for just a few minutes, and even in this brief interval, we see some apps with a very large number of connections to ad sites. With the exception of two applications – *Music Volume EQ* and *VidTrim - Video Trimmer* – none of the others are frequently downloaded (popular), and rated positively by users. Note that this information is missing for some of the apps that were removed from the PlayStore soon after we downloaded the APK for testing (and no further information is available).

Application	Trackers	Rate	Downloads
Eurosport Player	810	3.2	500.000
RunKeeper	804	4.4	10.000.000
Gestion du budget	725	4.3	500.000
Logo Quiz	301	4.5	10.000.000
Expedia Hotels et Vols	266	4.0	5.000.000
Vos Droits Quotidien	264	4.3	100
France TV Replay	261	2.8	10.000
Iron Man 3 Live Wallpaper	250	4.0	5.000.000
beIN SPORTS	236	3.6	500.000
NipCast	235	4.7	100

Table 9: Top 10 apps connecting to tracker URLs (italics indicate *Top Developer* status)

6.2 Tracking Intensity

We next examine the applications which connect to a large number of tracking services. As we observed from Fig. 2 the vast majority of mobile apps (73.2%) do not connect to tracker URLs. However, the ones that do connect to trackers tend to connect to a large number. The top 16% of the apps connect to 100 or more trackers. We note an interesting difference between the apps with prolific tracking and those that contact several ad sites. The "high-tracker" apps as shown in Table 9, tend to be overall more popular, highly rated and have not been removed fast from Google Play store, compared with the "high-adveristising" apps. To help support this argument, we note the incidence of the "Top Developer Badge" across these sets; Google awards these to developers based on some (opaque) combination of app design, trust and popularity. We find 4 of the 10 apps listed in Table 9 to be associated with this badge, while not a single app listed in Table 8 has it[2]. We surmise that this difference is mainly due to how the apps are monetized. The ad-driven apps are *directly* monetized by the ads they are displaying – and this tends to be intrusive to users. On the other hand, the apps with a high degree of targeting tend to be more embedded into the online ads ecosystem, and generate revenue *indirectly* by helping to construct profiles of the smartphone user, which can then be leveraged by them *and other apps and services*. However this deep integration with the ads ecosystem is difficult for an individual programmer, who has to resort to direct monetization (and which is likely to push users away).

6.3 App Suspiciousness

Finally we examine the *suspiciousness* of each application by leveraging the results from third party engines (as discussed previously). Intuitively, we would like an app to be suspicious when contacting many URLs that are tagged as being malicious (or at least, not benign), and *more* suspicious when these URLs are also

[2]Only 6% of the apps in our dataset are by developers with this certification.

spread over several domains. To this end, we define the following metric:

$$suspicion\_score = \sum_{i \in \mathbf{A}} (p_i^{\alpha}) \cdot d^{\beta} \quad (1)$$

where **A** is the set of URLs contacted by the app during its execution, p_i is the (absolute) number of positive (suspicious) signals from VirusTotal for the URL $i \in A$ (recall that the result, for each URL query, from VirusTotal is a vector of boolean signals). Finally, d is the distinct number of domains associated with URLs deemed suspicious by VirusTotal. The parameters $\alpha \geq 1, \beta \geq 1$ control the "weights" contributed by the suspicious URLs, or domain cardinality, to the score of the application. In our case, we set $\alpha = 3, \beta = 1$; while this is somewhat arbitrary, we note that increasing one (or both) parameters simply has the effect of affecting the relative scores between applications. In our initial effort, the suspiciousness score was found to be uniformly high across applications and this was attributed to connections made to very popular ad and tracking sites which were flagged by VirusTotal- *xiti.com*, *api.airpush.com*, *scorecardresearch.com*, *bluekai.com*, *ad.leadboltapps.net*. Whitelisting these popular (also manually verified to be legitimate) domains significantly improved the separation in scores across applications.

Table 10 shows the top 10 applications ordered by the suspiciousness score. Immediately, we see that *all* of these apps are associated with a low download count; in fact, we found several of these to have been since removed from the PlayStore. Examining these apps in detail, we found several instances where the app name is easily confused with a more well known app. We suspect that this "app name squatting" is deliberately meant to lure customers looking for the bonafide application. We discuss two examples from our top 10 list in greater detail – *PowerAMP Music Player (BASS)* and *Music Explorer*. The former is named suspiciously similar to the well known *PowerAMP* application (which has two developer badge awards). In fact, even the package is named to mislead the end-user (*installer.com.maxmpz.audioplayer*, vs. *com.maxmpz.audioplayer* for the bonafide application). The name "*Music Explorer*" has been used by several applications in the Google Play Store. Interestingly, the *Music Explorer* app listed as highly suspicious in Table 10, has been removed from the Google Store. In conclusion, using simple suspicion metrics (as the one in equation 1), we can easily blacklist apps, which can potentially harm the mobile device user.

6.4 App Category Behavior

In the previous sections, we focused on the behavior of particular applications. We next examine individual app categories with an aim to understand if there is some commonality of behavior among apps of the same category. Such a characterization is useful in understanding whether a certain application declared by its developer to be in a particular category, behaves in a manner consistent with apps

Application	Score	Rate	Downloads
PowerAMP Music Player	13203	4.0	10.000
Morandini Blog	9758	2.4	50.000
Exercice Machines Gym Demo	9464	4.0	10.000
Motor Racing News	7037	4.2	500
com.issakol12i.myapp	5825	-	-
Exercices Quotidien Fessiers*	4956	4.4	1.000.000
apps.buffalo.kmu.android	3480	-	-
Music Explorer	3366	4.4	50.000
biz.pompommanga.readcartoon	2920	-	-
Notability Basic Guide	2820	2.3	500

Table 10: Top 10 apps per suspicion score

of that kind (not suspicious), or whether its behavior shows marked differences compared to others (suspicious).

We start by looking at the overall number of HTTP connections made by different applications. Figure 3 shows a boxplot distribution of the number of URLs for each application, organized by category. Note that the outliers are indicated by '+' symbols in the graph. As seen in the figure, the one category that does stand out is *NEWS_AND_MAGAZINES*, where the median number of URLs connected to is about 150. At the other extreme, the median number of URLs connected to the *LIBRARIES_AND_DEMO* category is 1. While this seems to suggest that the category of an application strongly influences its chattiness, we do not find statistically significant differences across the remaining categories. In fact, the intra- category variation in this value seems to be higher than the inter- category variation. Specifically, although the majority of applications in each category connect to very few URLs, a significant number connect to a large number of URLs.

We extend our analysis, by further correlating the app categories with the type of URLs (ads and trackers) and the distinct domains they are connected to. Overall, we observe apps under *NEWS_AND_MAGAZINES* category to connect to higher number of distinct domains, trackers and ad URLs. However, similar to our previous observation, the differences among the remaining categories are not statistically significant, while the intra- category variations can be high. These results show that current information (such as app category) exposed to users in the Google Store is insufficient, and methods such as ours can provide valuable additional context to users.

We finally look to whether there are strong associations between app categories and the categories of domains communicated with. Table 11 presents the breakdown, in the domains being connected to, across various domain categories for *each* application category. Each row summarizes a particular set of applications in a category, and each column represents a particular domain category. Each table element indicates the fraction of domains related to a particular app category that belong to a specific domain category. For ease of representation, we only present the top 10 categories across all destination domains. Overall, we find that the 3 most popular domains relate to (1) advertisements (or domains related to online

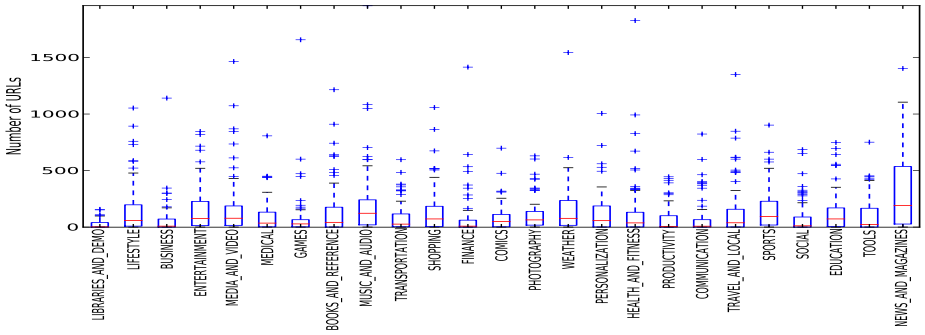


Figure 3: Number of URLs for different app categories

advertising), (2) search engines, and (3) information technology[3], (respectively). These three together account for about 60% of all the domains connected to. Apart from the above dominant domain categories, we observe other popular domain categories for particular app categories. For example, for *SHOPPING* apps, the third most popular domain category (with 10.3% of the domains) is *SHOPPING*. For *NEWS_AND_MAGAZINES* apps, the third most popular domain category is *SOCIAL*. However, apart from small exceptions, there is no sufficient correlation between the app category and the domain category, to draw out a systematic fingerprint of the application according to these criteria. From a user perspective, this pleads for a systematic verification of applications, regardless of their categories.

7 Application Description

The results presented thus far clearly indicate that applications on the Google Play Store often connect to destinations that are not essential for the operation of the app itself. Furthermore, much of this communication is completely hidden from users. In some cases, the end-user would benefit from an awareness of this activity (e.g., for user-tracking communication). Our methodology so far uses of a VPN server to intercept traffic; this is not practical on phones in active use (the added delay would affect user experience).

To enable end-users to get this visibility into installed applications, we built an Android application that monitors traffic, using a local proxy, from various installed applications and presents the mobile user additional context about the nature of this traffic and the end points being connected to.

The high level architecture of our app, which we call NSA (NoSuchApp, in honor of a similarly acronymed monitoring agency) is shown in Fig. 4. We use `SandroProxyLib` to establish a local HTTP (and HTTPS) proxy. By installing its own certificates, the proxy is able to emulate a man-in-the-middle for SSL traffic, which it can then monitor. HTTP(S) requests that are routed through the proxy

[3]IT category is used as a catch-all term for technology-related websites, when a finer grained characterization is unavailable.

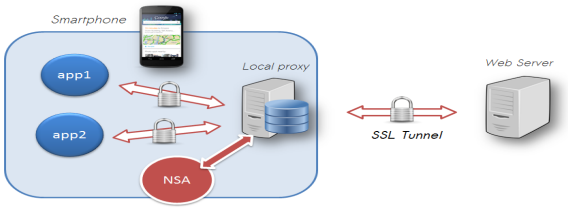


Figure 4: Application architecture: HTTP traffic is intercepted by a local proxy service

	DYN. CONTENT	MEDIA FILE DWNLD	FINANCIAL DATA	SHOPPING	ADS	TRAVEL	NEWS & MEDIA	IT	SEARCH ENGINES	SOCIAL
LIBRARIES/DEMO	4.9	1.6			26.2		6.6	31.2	19.5	
LIFESTYLE	7.1	0.2		1.6	23.2	1.6	4.1	24.3	15.1	4.6
BUSINESS	7.9				17.0		0.6	31.5	13.9	9.1
ENTERTAINMENT	7.8	0.5	0.3	0.8	20.1		3.7	26.2	10.7	2.9
MEDIA/VIDEO	6.0	0.2	0.2	0.5	26.0		5.2	25.1	10.3	5.1
MEDICAL	8.1				29.6		5.9	27.4	9.1	3.2
GAMES	6.4			0.8	30.1		2.8	29.1	9.8	0.3
BOOKS/REFERENCE	6.3	0.3		0.6	29.8		5.8	24.2	13.2	2.5
MUSIC/AUDIO	8.8	2.6		0.8	21.5		3.2	24.0	9.7	5.2
TRANSPORTATION	5.1		0.4	0.4	24.3	6.4	3.8	27.2	17.0	0.4
SHOPPING	8.7			10.3	10.9		2.2	25.8	8.2	9.5
FINANCE	6.6	1.5	8.8		21.2		2.2	31.4	8.8	5.4
COMICS	4.4				31.9		5.4	20.1	13.2	3.9
PHOTOGRAPHY	4.9	0.3		0.3	30.5		4.9	20.1	15.2	0.9
WEATHER	4.8	0.3		0.3	26.8	1.3	9.1	25.0	14.4	3.0
PERSONALIZATION	8.2			0.7	21.0		6.1	28.4	17.3	0.4
HEALTH/FITNESS	5.7			0.4	27.8	0.4	3.6	25.8	15.2	4.9
PRODUCTIVITY	9.6				27.0		5.7	27.8	13.0	3.0
COMMUNICATION	3.5			1.3	28.0		6.1	25.3	17.0	2.6
TRAVEL/LOCAL	2.9		0.3	0.5	20.3	8.8	4.0	21.3	19.5	3.5
SPORTS	5.4	0.3			18.3		3.0	24.3	11.4	9.9
SOCIAL	4.8	0.4			16.7	1.1	3.7	30.1	14.5	5.2
EDUCATION	3.8			0.6	32.6		4.1	23.1	17.1	3.5
TOOLS	1.9				33.6	0.5	7.0	27.6	14.5	1.9
NEWS/MAGAZINES	3.0	0.2	0.2	0.8	19.5	0.2	8.4	30.6	10.3	11.2

Table 11: App categories vs. app domain categories. (Data is in percentage (%))

(which is the case for most applications that use the default network settings) trigger callbacks in our application, which then logs and classifies the HTTP(S) connections. While designing NSA, we had to overcome two challenges – (i) attributing flows to applications correctly, and (ii) classifying the destination URLs without too much overhead. To solve the first challenge, NSA periodically polls the system structures in *proc/net/tcp* and *proc/net/udp* (also the structures supporting IPV6), and extracts the mapping between application UID and the open sockets (ip address and port). By correlating the socket information with the URLs seen by the proxy, we correctly associate applications with the URLs being generated. To solve the second challenge, we batch a number of URLs together before issues queries to various online engines to amortize resource usage (CPU, network). Using our application, users can view all the destination end-points associated with a particular application that correspond to one of three categories: (i) third party trackers, (ii) suspicious websites[4], and (iii) destination addresses for which the proxy is bypassed (while there are valid reasons to do so, this is likely also the behavior of malicious applications).

With this application, our goal is to provide a mechanism for end-users to be aware of the network activity of their installed Android applications. All of the Android users among the authors have been running the application for several weeks, and on one phone (which was previously installed with a rootkit), the application helped identify traffic from applications (that were automatically included with the rootkit) to suspicious destinations (even though the applications were never launched by the user in question). The application is available as an installable package at `https://db.tt/Cx8fB5Xz`. We plan to make the app publicly downloadable via the Google Play store in the near future.

Looking much further, we can envision a crowdsourced app reputation system driven by NSA where individual users can inspect the traffic being generated by applications tag is as being normal, or else unexpected, or suspicious. Such individual signals could be aggregated at a backend and fed back into the application. This would enable easy blacklisting of applications (and their traffic) based on what other users have observed and reacted to. The exact design of such a crowdsourced system is outside the scope of this paper, and we hope to realize it in future work.

8 Conclusion

The lack of oversight in Android Play Store makes it all too easy for end-users to install applications of dubious origin, or those which silently carry out activity that might not be seen favorably by the user. In this paper, we describe a lightweight characterization methodology that can generate descriptions of application network behavior in an automated manner. The descriptions shed light on the nature of the

[4]We currently rely on the characterization provided by the Google Safe Browsing API, but will eventually use the analysis from VirusTotal.

websites communicated with, focusing on those that may be undesirable to the end-user (ad related, tracking, and malicious).

Using this methodology, we conduct a characterization study of a large number of applications from the Google Play Store. As a general comment we confirm that applications in any domain may carry undesirable activity. This stresses the need for using broad and diversified datasets rather than focussing on specific categories of applications. In addition, our results reveal several interesting insights: (i) that a significant number of applications, some highly rated, download an excessive number of advertisements which indicate that users may not be as sensitive to advertisements as anecdotally conjectured; (ii) a large number of applications communicate with a multiplicity of online tracking entities, a fact to which users may not be aware; and (iii) we find some applications communicating with websites that have been deemed malicious by malware detection engines. Our results underscore the need for greater transparency in the network interaction of mobile applications on the Android App store(s). To this end, we also describe the design of our own application that provides exactly this service. With our application, end-users are able to understand the different domains the application is communicating with which enables them to make informed decisions about the desirability of the applications they install.

References

[1] AdBlock Plus. `https://adblockplus.org/`.

[2] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. In *PLDI*, 2014.

[3] S. Chakraborty, C. Shen, K. R. Raghavan, Y. Shoukry, M. Millar, and M. Srivastava. ipShield: A Framework for Enforcing Context-Aware Privacy. In *NSDI*, 2014.

[4] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner. Analyzing inter-application communication in android. In *MobiSys*, 2011.

[5] L. P. Cox, G. P., L. G., P. V., R. A., and C. S. Wu B. Spandex: Secure password tracking for android. In *USENIX*, 2014.

[6] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song. NetworkProfiler: Towards automatic fingerprinting of Android apps. In *INFOCOM*, 2013.

[7] Do Not Track: Universal Web Tracking Opt Out. `http://donottrack.us`.

[8] M. Egele, C. Kruegel, E. Kirda, and G. Vigna. Pios: Detecting privacy leaks in ios applications. In *NDSS*, 2011.

[9] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In *OSDI*, 2010.

[10] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A first look at traffic on smartphones. In *IMC*, 2010.

[11] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *WISEC*, 2012.

[12] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall. These aren't the droids you're looking for: Retrofitting android to protect data from imperious applications. In *CCS*, 2011.

[13] P. Leon, B. Ur, R. Shay, Y. Wang, R. Balebako, and L. Cranor. Why Johnny Can't Opt Out: a Usability Evaluation of Tools to Limit Online Behavioral Advertising. In *ACM CHI*, 2012.

[14] L. Li, A. Bartel, J. Klein, Y. L. Traon, S. Arzt, S. Rasthofer, E. Bodden, D. Octeau, and P. McDaniel. I know what leaked in your pocket: Uncovering privacy leaks on android apps with static taint analysis. Technical Report ISBN 978-2-87971-129-4, University of Luxembourg, April 2014.

[15] L. Lu, Z. Li, Z. Wu, W. Lee, and G. Jiang. Chex: statically vetting android apps for component hijacking vulnerabilities. In *CCS*, 2012.

[16] G. Maier, F. Schneider, and A. Feldmann. A first look at mobile hand-held device traffic. In *PAM*, 2010.

[17] D. Octeau, P. McDaniel, S. Jha, A. Bartel, E. Bodden, J. Klein, and Y. Le Traon. Effective inter-component communication mapping in android with epicc: An essential step towards holistic security analysis. In *Usenix Security*, 2013.

[18] M. M. Tsang, S.-C. Ho, and T.-P. Liang. Consumer attitudes toward mobile advertising: An empirical study. *International Journal of Electronic Commerce*, 8(3), 2004.

[19] B. Ur, P. Leon, L. Cranor, R. Shay, and Y. Wang. Smart, Useful, Scary, Creepy: Perceptions of Online Behavioral Advertising. In *SOUP*, 2012.

[20] VirusTotal Meta Engine. `https://www.virustotal.com/`.

[21] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos. Profiledroid: Multi-layer profiling of android applications. In *Mobicom*. ACM, 2012.

[22] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman. Identifying diverse usage behaviors of smartphone apps. In *IMC*, 2011.