Whom Does Your Android App Talk To?

Xuetao Wei[‡] Iulian Neamtiu[*] Michalis Faloutsos[*]
[‡]University of Cincinnati [*]University of California, Riverside
xuetao.wei@uc.edu neamtiu@cs.ucr.edu michalis@cs.ucr.edu

***Abstract*—Smartphone privacy and security work has focused mostly on malicious apps. We take a different angle by questioning whether good apps suffer from a lack of judgment and interact with "bad" websites. We use the term bad websites to refer to entities that engage in dangerous or annoying activities that range from distributing malware, to phishing and overly aggressive ad spamming. The focus of our work is this relatively neglected aspect of security: "Whom does an app talk to?" In this paper, we design and implement** AURA**, a framework for identifying the hosts that an app talks to and evaluating the risks this communication entails.** AURA **makes use of both static and dynamic analysis. We studied 13,500 popular free Android apps that connect to 254,022 URLs and 1,260 malicious Android apps that connect to 19,510 URLs. Our main contribution is showing that good apps pose security risks as they contact at least one website that: (a) distributes malware (8.8% of apps), (b) are in a blacklist (15% of apps) based on the classification by VirusTotal and Web of Trust. Our work can raise awareness that even good apps need to be carefully evaluated, especially as people become more concerned about smartphone security and privacy.**

I. Introduction

Apart from portability and form factor, the popularity of the smartphone platform is due to apps that are providing more and more essential functions. At the same time, the apps collect more information about us with sensors that offer a wide range of context-sensitive functionality, from GPS- and compass-assisted navigation to song recognition to exercise tracking to picture geo-tagging and sharing. This is illustrated in Figure 1: the private data (e.g., location, phone state, list of contacts) can be leaked by apps to all kinds of websites:[1] good, bad, or somewhere in-between.

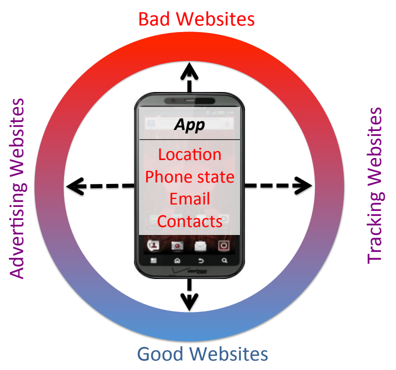


Fig. 1. Even good apps communicate with websites of variable reputation, which can raise a range of concerns.

We focus on a specific security and privacy question: *Do good apps talk to "bad" websites?* We use the term good apps here loosely to refer to apps that come from reputable developers, and widely vetted by large numbers of users. For the term bad websites, we use it to refer to hosts and domains that have been labeled as inappropriate by malware repositories such as VirusTotal [3] and trustworthy reputation engines such as Web-Of-Trust (WOT) [4]. In general, these bad websites engage in dangerous or annoying activities that range from distributing malware, to phishing to overly aggressive ads and spamming, as we discuss in Section III. Adopting the terminology from WOT, we define the terms: (a) malicious website, implicated in distribution of malware, (b) bad website, that appears in blacklists, and (c) low-reputation websites that have a user rating of less than 60, with each category including the previous in the order presented. For labeling a website, we rely on information and the lists from WOT and VirusTotal, which are widely-used and widely considered as reference sources.

We focus on apps' network communication, since it is an obvious vector for security attacks: Internet access is a *de facto* capability for almost all apps. On the Android platform, most apps request Internet permission, while all apps in the iOS App Store have Internet access by default without even asking the user. It is also highly unlikely that apps will refrain from getting Internet access: (a) many apps needs access to the Internet to operate, and (b) many apps, especially the free ones, seek revenue by either showing ads or collecting information about the user and her behavior, such as mobility patterns. So the question is: does Internet access pose concerns for security and privacy, even for good apps? Interestingly, even identifying which websites an app talks hides several subtleties.

In this paper, we propose AURA (Android Url Risk Assessor), a systematic approach to identifying security and privacy concerns for apps based on the websites that an app talks to. The first step is to comprehensively identify all such websites for a given app, which is non-trivial. We propose and compare the use of both dynamic and static analysis, and we argue that static analysis is necessary as many embedded websites are not contacted, even during exhaustive test runs. Second, we provide a taxonomy of websites using both malware detection and crowdsourcing efforts to capture a wide range of annoying or dangerous activities. We study 13,500 popular free Android apps from Google Play [1] that connect to 254,022 URLs and 1,260 malicious Android apps [21] that connect to 19,510 URLs.

The results of our work can be summarized in the following points.

a. Developing AURA. We develop a systematic and comprehensive approach focusing on a lesser-studied security aspect of apps, which uses both static (bytecode) analysis and dynamic (execution) analysis when available. We employ widely-used classification labels for the bad websites, in order to make reporting of results consistent with industry standards.

[1]Websites, domains, hosts and entities are used interchangeably in our paper.

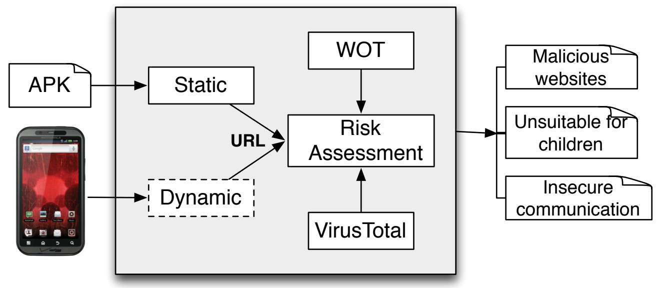


Fig. 2. AURA architecture.

b. The importance of static analysis. We show that dynamic analysis cannot match the thoroughness of static analysis. Even when apps are explored thoroughly (on average for two hours each) using high-coverage automated tools, dynamic analysis identifies less than half the URLs static analysis does. This suggests that using static analysis provides significant insight into an app's potential communication.

c. Good apps talk to bad websites. We find that good apps can be interacting with questionable websites: for our examined apps, 8.8% communicate with malicious websites, 15% talk to bad websites, 73% with low-reputation websites (as defined above), and 74% of the apps talk to websites containing material not suitable for children.

d. Understanding bad intentions. We find the following intentions of bad websites: 43% of bad websites try to phish sensitive personal information or confidential financial information, e.g., credit card details, while 42% of bad websites are used for distribution of rootkits, trojans, viruses, malware, spyware, rogues and adware, and creating virus attacks. The rest of the bad websites, which account for 15%, intrusively and aggressively sell ads. These intentions vary from harming devices to stealing confidential data to annoying users.

Potential uses and deployment: We envision using AURA in several different ways.

a. Advisory stand-alone tool. AURA could be used as an advisory stand-alone tool, where users submit the apps of interest, and receive an assessment; the set of users includes researchers that want to further study app security from an Internet access point of view.

b. Expanded app information. AURA could enhance the information presented to an user prior to installing an app. The Google Play market information panel could include AURA's assessment as a part of the profile of the app as a more refined explanation of the Internet Access permission.

c. App filtering. AURA could also be used as a filter before the app is allowed to enter Google Play. The market owner, such as Google, Samsung or Amazon, could force developers to evaluate their apps with AURA, and allow apps on the market only if they meet certain requirements (not talking to malware-hosting sites seems like a good requirement).

d. A component in a larger security system. AURA could be integrated into other static and dynamic analysis tools to provide more comprehensive risk information for each app. The interactions between the developer and market administrators are encouraged during the development and maintenance of the app.

Limitations: Our primary focus is to detect risk associated with good, well-intended apps which contain URLs to malicious or simply questionable domains. On the flip side, a malicious app developer who wants the app to contact a bad website has a lot of opportunity to hide and make the detection difficult. For example, one can obfuscate the website name, making it difficult to detect through static analysis; or create indirections, perhaps even use fast flux networks, where the redirection target varies dynamically over time. Nevertheless, AURA was capable of discovering malicious websites in malicious apps, as discussed in Section III-B.

II. Approach

We now proceed to presenting our approach by discussing each component of AURA as shown in Figure 2. Determining the URLs an app talks to is challenging. Static analysis (in this context) means extracting the URLs embedded in the app by analyzing the app bytecode. Dynamic analysis means observing the URLs accessed by the app dynamically, by instrumenting the app or the smartphone. We performed a preliminary investigation into the suitability of each of these techniques by comparing the statically-discovered URLs with dynamically-discovered URLs. We found that, even when driven by systematic exploration tools, dynamic analysis misses many more URL compared to static analysis for all the selected apps. Hence for our large-scale investigation on the set of 13,500 apps we used static analysis only.

A. Static Analysis Component

An Android app is distributed as an `.apk` file which contains the compressed bytecode of the app (a `.dex` file) along with app resources. For each app, we decompiled the `.dex` file, and scanned it to find the URLs embedded inside the app. Next, we extract the domains from these URLs. We need to be clear that a domain is different from a URL. One URL could be one domain directly (e.g., www.facebook.com) or only one resource path in one domain (e.g., www.facebook.com/login). Multiple URLs may have the same domain.

Note that static analysis could miss URLs when websites use redirection, dynamic content, JavaScript, etc. However, we argue that our method achieves a very good approximation of the number of entities the app talks to. The domains we find are the least domains ("lower bound") that the app will talk to. As discussed shortly (Section II-C and Table I), we have observed that our static method works significantly better than just dynamically running apps. In AURA, we envision that static analysis and dynamic analysis should work together. If extra domains are found during app usage, the dynamic analysis component could complement the static analysis component, as we will see in Table I.

B. Dynamic Analysis Component

Dynamic analysis requires a set of inputs or tests to drive execution. Unfortunately, relying on users to manually produce such suites is ineffective. First, previous studies show that it is difficult and expensive to dynamically analyze apps as they run on the phone [20]. Second, other studies have shown that even when combining usage data from multiple users, only about

30.08% of an app's screens and 6.46% of an app methods are explored [17]. To alleviate the manual burden yet achieve good coverage, we used $\mathsf{A}^3\mathsf{E}$, a tool which automatically explores apps in a systematic way [17]. $\mathsf{A}^3\mathsf{E}$ automatically generates the events to test each app thoroughly (sometimes over several hours) to cover on average around 60% of the screens and 33% of the methods, which is far superior to manual efforts. Finally, during execution, we use `tcpdump` to collect the traffic data and identify the traffic entities.

C. *Why is Static Analysis Preferable?*

Dynamic analysis critically hinges on the availability of a good (high-coverage) testing suite, so that all the facets of an app can be explored. As argued above, even sophisticated tools such as $\mathsf{A}^3\mathsf{E}$ have limited reach in terms of how thoroughly an app is explored. In the following, we demonstrate the effectiveness of static analysis when compared with dynamic analysis. In order to ensure representative results, we selected our test apps by following rigorous criteria as we did in previous work [20]. In Table I, we present a comparison of the two analyses for each app, and over all apps. The first column contains the app name. The second column shows the exploration time required by dynamic analysis. Note that dynamic exploration is thorough, with apps being explored on average for *two hours*, which is far longer than the typical average app user session (71.56 seconds) [7]. The third column shows the total number of URLs discovered by dynamic analysis for each app; on average, 6.1 URLs per app. The fourth column shows the number of URLs discovered via dynamic analysis that could not be found via static analysis. The last two columns show the number of URLs discovered by static analysis (total and the extra URLs compared to dynamic analysis). We observe that static analysis finds on average 11.9 domains that dynamic analysis does not find, whereas dynamic analysis finds on average 2.8 domains that static analysis does not. Therefore, we chose static analysis as our preferred method for performing the rest of the study. An additional advantage of static analysis is scalability: as URL extraction and classification takes on the order of seconds per app, static analysis is particularly suitable for analyzing large sets of apps.

App	**Dynamic analysis**			**Static analysis**	
	Time (minutes)	Total URLs	Extra URLs	Total URLs	Extra URLs
Amazon	131	5	2	10	6
AdvncdTaskKiller	47	0	0	3	3
AdvncdTaskKiller($)	58	0	0	0	0
BBCnews	52	13	8	5	2
CNN	161	8	6	12	10
Craigslist	91	7	3	7	3
Dictionary.com	131	19	15	13	8
Dictionary($)	156	0	0	13	13
Dolphin Browser	179	6	0	14	12
ESPN	44	1	1	7	7
Flixster	219	9	4	20	16
IMDB	126	4	3	17	17
InstantHeartRate	51	0	0	27	27
InstantHeartRate($)	49	0	0	27	27
Pandora	111	3	0	19	19
Picsay	121	0	0	3	3
Picsay($)	129	0	0	3	3
Shazam	239	12	4	24	20
Shazam($)	230	12	4	24	20
Weatherbug	107	14	3	16	16
Weatherbug($)	124	14	1	16	16
ZEDGE	114	8	8	16	15
Average	*121*	*6.1*	*2.8*	*13.4*	*11.9*

TABLE I. SUMMARY OF STATIC AND DYNAMIC ANALYSES; "EXTRA" REFERS TO DOMAINS FOUND BY ONE ANALYSIS BUT NOT THE OTHER.

D. *Classification and Evaluation*

In this section, we try to demystify the usage of network entities and evaluate the risks from both technical and crowd-sourcing perspectives. Typically, antivirus software might protect the device or app when it comes to viruses and malware, but dangers such as scams, phishing and untrustworthy online stores are hard to detect by traditional technical methods. Furthermore, it is non-trivial to classify an app as malicious, privacy infringing, or benign. In addition, a recent study from McAffee, a major industrial security company, shows that the dividing line between benign and malicious apps is not so clear and there are increasing numbers of risky (albeit not malicious) apps in app stores [2]. AURA aims to address this. As we mention above, we demystify the usage of URLs and domains and use two-level analysis (both technical and crowdsourcing approaches) to evaluate the risks associated with the URLs. We first use traditional technical methods (the VirusTotal database) to scan the URLs to determine whether the URL is malicious. Next, we further evaluate the domains the app talks to. We then use the Web of Trust (WOT), a crowdsourcing approach, to determine the nature (e.g., scam, phishing, malware hosting) and reputation of domains. WOT is widely used by many enterprises, e.g., Facebook; it is based on a crowdsourcing approach that aggregates feedback from a global community of millions of users who rate and comment on domains based on their own experiences [4]. WOT allows users to rate a domain in terms of trustworthiness and child safety; then it aggregates user ratings as well as information from other sources into a rating along each of the dimensions as well as a combined score, from 0 to 100. Low reputation is defined by WOT as a website having a reputation score < 60, and this low reputation is assigned due to: bad customer services, misleading advertising, distributing malware and phishing users' information, etc.

Using WOT enables AURA to reveal the threats that only humans can spot (via crowdsourcing), such as scams, unreliable web stores and questionable content, which is especially valuable as we want to understand who is using our personal information. This complements the traditional security solutions that protect smartphones against technical threats such as viruses and other harmful software [4]. Detailed analysis results for both URL and domains can be found in Section III

III. Evaluation Results

Our study is based on a sizable number of apps, both benign and malicious. After we apply AURA to these apps, we obtain 254,022 URLs from 13,500 benign apps and 19,510 URLs from 1260 malicious apps. In the remainder of the paper, with the exception of Section III-B where we discuss malicious apps, all the findings are based on analyzing the 254,022 URLs that the 13,500 benign apps talk to [16]. Note that the total number of URLs was obtained by adding the number of URLs for each app, so the dataset contains duplicates; we will discuss this next.

1	admob.com	11	tapjoyads.com
2	android2020.com	12	mydas.mobi
3	twitter.com	13	adwhirl.com
4	facebook.com	14	w3.org
5	airpush.com	15	wikipedia.org
6	google.com	16	amazonaws.com
7	android.com	17	psesc.com
8	gstatic.com	18	inmobi.com
9	mobclix.com	19	paypal.com
10	flurry.com	20	hubblesite.org

TABLE II. TOP 20 DOMAINS USED IN APPS.

A. Malicious URLs

Malicious sites could be visited or interacted with when using Android apps, which poses great risk especially when more and more apps harvest the personal information stored on the smartphone. We examined whether malicious URLs are used in our benign app dataset. To find such URLs, we cross-checked against VirusTotal's database. We found that 286 URLs were malicious; these URLs spread across 1,187 apps (8.8%). We believe that the 8.8% percentage is a significant source of concern. In Section III-C we present a detailed analysis of these blacklisted domains.

We now turn to investigating the domains (or hosts) in our dataset, that is the trustworthiness of sites without regard to the specific path. We found that 66% of the apps talk to at least one domain that has very poor reputation; that 74% of the apps talk to websites containing material not suitable for children; that malicious apps do not necessarily talk to ill-reputed websites; and that 15% of apps talk to blacklisted domains. We also found that Android apps tend to have more tracking services than advertisement services.

We also computed the top domains used in the 13,500 benign Android apps, and present the top-20 in Table II. We can see that advertisements (e.g., admob.com, flurry.com, airpush.com, inmobi.com), cloud services (e.g., amazonaws. com), social networking services (e.g., twitter.com, facebook. com) and payment solution services (e.g., paypal.com) are used intensively among apps. We also plot the distributions about the number of tracking and advertisement services of each app in Figure 3(a) and Figure 3(b). We can see that both services penetrate Android apps broadly, while Android apps tend to have more tracking services than advertisement sources.

B. Trustworthiness and Child Safety

We now investigate the reputation of the domains extracted from our URL dataset with respect to trustworthiness and child safety. WOT assigns each domain a reputation score from 0 (very poor reputation) to 100 (very good reputation). We use WOT's domain reputation score to compute the reputation for each domain in our dataset. If we could not find the reputation for a domain in WOT's database, we just assign that domain a reputation score of -1. For each app, we compute several reputation indicators: a *minimum reputation*, that is the lowest reputation score across all the domains used in the app; an *average reputation*, that is the average reputation score across all the domains used in the app; and a *reputation span*, that is the difference between the minimum and maximum reputation score across all the domains used in the app;

Benign apps: Our analysis has revealed several reasons for concern. We found that 63% of apps talk to at least one domain for which WOT does not have any reputation score. In addition, 73% of the apps talk to at least one domain that has unsatisfactory reputation (score is less than 60). In detail, 68% of the apps talk to domains with poor reputation (score is less than 40) and 66% of the apps talks to the domains with *very poor reputation* (score less than 20). Unsurprisingly, these trends are also reflected in the reputation span: 60% of the apps have reputation spans exceeding 90 points, meaning the apps mix high-reputation with low-reputation domains. These findings indicate that there is significant cause for concern even for benign apps: when these supposedly benign apps send information to low-reputation domains, users can be exposed to privacy and security risks.

Children use Android apps for many activities, e.g., gaming or social networking. Hence we proceed to analyze how child-safe the domains are, based on WOT's definition of child-safe domains. Similar to trustworthiness, we plot the child-safety reputation score for benign apps in Figure 3(e). We observe that 74% of the benign apps talk to at least one domain that has unsatisfactory reputation based on user ratings, hence may not be suitable for children (for example, the website contains adult material). We believe that such findings could help Google Play, the main Android app marketplace, to better regulate app distribution in order to safeguard child safety.

Malicious apps: Intuitively, we would assume that malicious apps would contain low-reputation of domains. We do the same reputation evaluation for the malicious apps, and we find that our intuition would be wrong—malicious apps have similar distribution of trustworthiness and child safety as benign apps. For example, in terms of trustworthiness Figure 3(d) indicates that about 55% of the apps have reputation less than 15, and there are fewer apps with large reputation spans. Child safety reputations (Figure 3(f)) are also similar to benign apps.

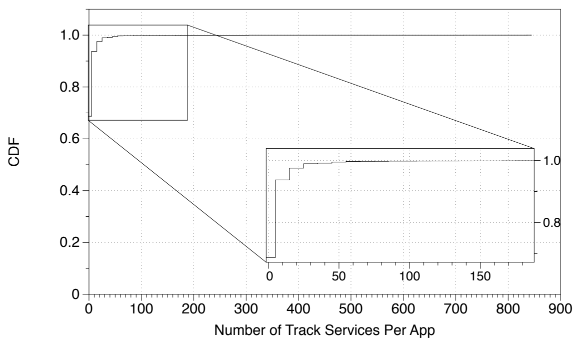
This is unsurprising, since most malicious apps are created by injecting a malware veneer in a benign app via repackaging [23]. The edge that AURA provides is the ability to examine the reputation of domains the app talks to: it is important to tackle Android app security not only via traditional security techniques (that protect devices against technical threats such as viruses and other harmful software), but also via crowdsourcing. Hence our AURA approach can help protect the device and the app against threats that only the human eye can identify, such as scams, unreliable web stores and questionable content.

Blacklist type	Description
malware	Site is blacklisted for hosting malware
phishing	Site is blacklisted for hosting a phishing page
scam	Site is blacklisted for hosting a scam

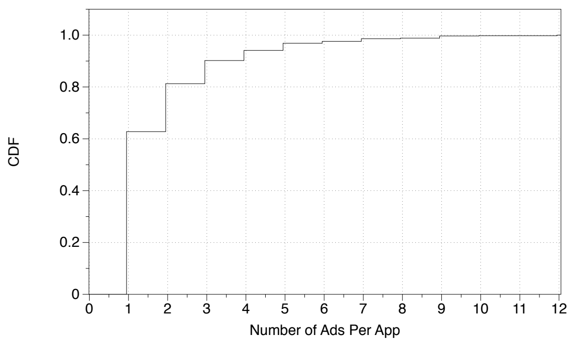
TABLE III. DEFINITIONS AND DESCRIPTIONS OF BLACKLISTED TYPES OF DOMAINS FROM WOT.

C. Blacklisted Domains

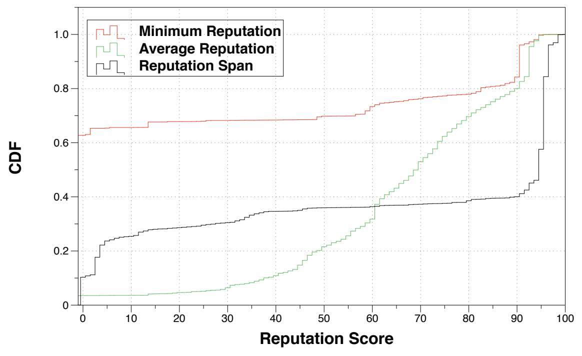
Blacklisted domains are known for hosting malwares or viruses, phishing and scam hosts, as shown in Table III. Surprisingly, according to both VirusTotal and WOT'ratings, AURA found that 2,025 apps (15% of the dataset) talk to blacklisted domains. These blacklisted domains pose a wider rage of dangers to end-users, e.g., users' sensitive data could be leaked to these domains for illegal purposes, or users could end up downloading and installing malware. We provide a detailed



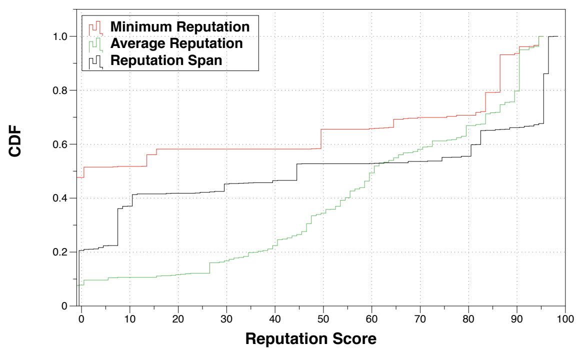
(a)



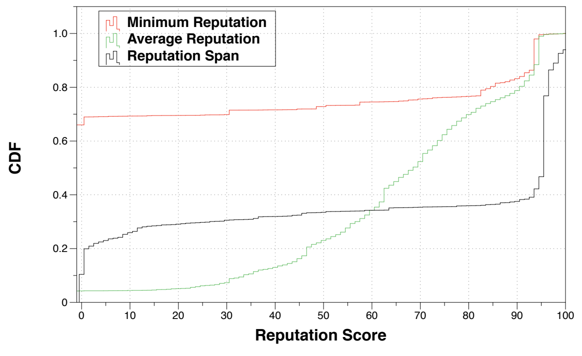
(b)



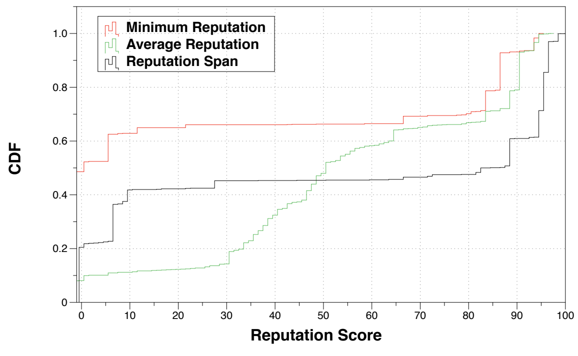
(c)



(d)



(e)



(f)

Fig. 3. 3(a): Distribution of number of tracking services per each app; 3(b): Distribution of number of advertisement services per each app; 3(c): Domain reputation distribution in benign apps; 3(d): Domain reputation distribution in malicious apps; 3(e): Child safety reputation distribution in benign apps; 3(f):Child safety reputation distribution in malicious apps.

Category	%
Advertising	28
Hosting Service	23
Entertainment Media	14
Short URL Service	8
Dating	7
Social	7
DNS service	6
Online Shopping	3
Finance	2
Misc	2

TABLE IV. CATEGORIES OF BLACKLISTED DOMAINS AND THEIR PERCENTAGES.

break-down of blacklisted site categories in Table IV. As we can see, they include advertising services, hosting services, financial services, etc. We have manually browsed some of the blacklisted domains to discover how they lure users in and how they exploit users and there information. We provide details on their nefarious behavior next.

1) Luring Users In: Blacklisted websites first use a "lure-in" to entice users into visiting the website or clicking on download links, namely, using these means:

1) "Big reward" return trap: cheat users by claiming that they could obtain a big return after they buy the advertised promotions from the website.
2) Adult content: using explicit images to lure users into subscribing to services.
3) Intrusive ads, that is display ads that constantly pop up, counting on user's attrition to eventually click on the ad.
4) Fake sites: present a deceiving front page (e.g., news, government, bank, travel) to lure users into sharing their sensitive and confidential information.
5) Exploited sites: sites or hosting services that are compromised by malware.
6) Abusing short URL and DNS services: using an

URL shortener to hide their suspicious intentions and redirect the users to malicious sites.

2) Inflicting Malicious Behavior: Once users are lured into visiting websites, sharing information, or downloading software, the blacklisted websites exploit users' good faith by inflicting malicious behavior in a variety of ways, which we describe next.

Phishing sensitive personal information or confidential financial information, e.g., credit card details. Once users share confidential information, websites will resort to identity theft, credit card abuse, and tracking users' habits.

Distribution of rootkits, trojans, viruses, malware, spyware, rogues and adware, and creating virus attacks. Once such nefarious software is installed, the malicious behavior can take a variety of forms: corrupting the data saved on the smartphone, which could render the phone unusable, information leaking (e.g., financial information, passwords), and so on.

Intrusively and aggressively sell ads. Once such adware is installed, it displays non-stopping pop-up ads that users cannot dismiss/unsubscribe from.

Note that a blacklisted domain may have two or more of these behaviors, which means some of these intentions can co-exist.

IV. Related work

Most security studies focus on malicious apps or OS-level exploits of good apps. There are several studies that focus on identifying malicious apps [12], [21], [22], analyzing the source code and the OS behavior and permissions [5], [6], [8], [15], [16]. Then, there is a group of studies that study network traffic patterns [9], [14], [20]. Other efforts focus on user information leakage and attempt to detect the specific information that is being leaked [11], [13], [18]. Systems that rely on instrumenting the whole software stack, e.g., TaintDroid, can warn users when an Android app leaks sensitive data over the network [19]. The developer can also use the system OASIS to have control on the data that the applications are using [10]. Finally, there are web-oriented efforts, not necessarily focusing on smartphones, that evaluate and label websites [3], [4] either focusing on malware, or considering a wider range of goodness based on user-feedback; we leverage such efforts to classify the websites here. However, our focus is different, in that we want to find out: (1) Which entities is the personal data potentially sent to (e.g., content providers, advertisers)? (2) Are these entities trusted? These questions are not answered in prior work.

V. Conclusion

We have presented AURA, an approach and tool that can perform scalable risk assessment of the hosts Android apps communicate with via static and/or dynamic analysis. Experiments with using AURA on 13,500 popular benign Android apps have revealed that a sizable share of good apps talk to bad websites: these apps communicate with websites that have low reputation, or outright host malware, or are unsuitable for children. We believe that the scalable URL risk assessment offered by AURA can potentially benefit Android app users, developers and researchers.

Acknowledgements

This work was supported in part by funds from the University of Cincinnati CECH, as well as National Science Foundation awards CNS-1064646, NSF SaTC 1314935, and FA8650-15-C-7565. 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 National Science Foundation or the U.S. Government.

References

[1] Google Play. https://play.google.com/store, September 2013.

[2] McAfee. http://www.mcafee.com/us/resources/reports/rp-mobile-security-consumer-trends.pdf, 2013.

[3] VirusTotal. https://www.virustotal.com/en/#url, Sept. 2013.

[4] Web of Trust. http://www.mywot.com/, Sept. 2013.

[5] A.P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner. Android Permissions: User Attention, Comprehension, and Behavior. In *SOUPS*, 2012.

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

[7] M. Böhmer, B. Hecht, J. Schöning, A. Krüger, and G. Bauer. Falling asleep with angry birds, facebook and kindle: a large scale study on mobile application usage. In *MobileHCI '11*, pages 47–56.

[8] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner. Analyzing Inter-Application Communication in Android. In *ACM MobiSys*, 2011.

[9] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A First Look at Traffic on Smartphones. In *ACM IMC*, 2010.

[10] M. Conti, E. Fernandes, J. Paupore, A. Prakash, and D. Simionato. OASIS: Operational Access Sandboxes for Information Security. In *ACM SPSM*, 2014.

[11] M. Egele, C. Kruegel, E. Kirda, and G. Vigna. PiOS: Detecting Privacy Leaks in iOS Applications. In *NDSS*, 2011.

[12] M. Grace, Y. Zhou, Q. Zhang, S. Zou, X. Jiang. RiskRanker: Scalable and Accurate Zero-day Android Malware Detection. In *ACM MobiSys*, 2012.

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

[14] Q. Xu, J. Erman, A. Gerber, Z. Morley Mao, J. Pang, and S. Venkataraman. Identify Diverse Usage Behaviors of Smartphone Apps. In *IMC*, 2011.

[15] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones. In *NDSS*, 2011.

[16] S. Fahl, M. Harbach, T. Muders, L. Baumgartner, B. Freisleben, and Matthew Smith. Why eve and mallory love android: an analysis of android SSL (in)security. In *ACM CCS*, 2012.

[17] T. Azim and I. Neamtiu. Targeted and Depth-first Exploration for Systematic Testing of Android Apps . In *OOPSLA*, 2013.

[18] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri. A Study of Android Application Security. In *USENIX Security Symposium*, 2011.

[19] 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.

[20] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos. ProfileDroid: Multi-layer Profiling of Android Applications . In *ACM MobiCom*, 2012.

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

[22] Y. Zhou, Z. Wang, Wu Zhou and X. Jiang. Hey, You, Get off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets . In *NDSS*, 2012.

[23] W. Zhou, Y. Zhou, X. Jiang, and P. Ning. Detecting repackaged smartphone applications in third-party android marketplaces. In *CODASPY*, pages 317–326, 2012.