AndRadar: Fast Discovery of Android Applications in Alternative Markets

Martina Lindorfer[1], Stamatis Volanis[2], Alessandro Sisto[3],
Matthias Neugschwandtner[1], Elias Athanasopoulos[2], Federico Maggi[3],
Christian Platzer[1], Stefano Zanero[3], and Sotiris Ioannidis[2]

[1] Secure Systems Lab, Vienna University of Technology, Austria
{mlindorfer,mneug,cplatzer}@iseclab.org
[2] Institute of Computer Science, Foundation for Research & Technology – Hellas, Greece
sebolani@gmail.com,{elathan,sotiris}@ics.forth.gr
[3] Politecnico di Milano, Italy
alessandro.sisto@mail.polimi.it,{federico.maggi,stefano.zanero}@polimi.it

Abstract. Compared to traditional desktop software, Android applications are delivered through software repositories, commonly known as application markets. Other mobile platforms, such as Apple iOS and BlackBerry OS also use the marketplace model, but what is unique to Android is the existence of a plethora of alternative application markets. This complicates the task of detecting and tracking Android malware. Identifying a malicious application in one particular market is simply not enough, as many instances of this application may exist in other markets. To quantify this phenomenon, we exhaustively crawled 8 markets between June and November 2013. Our findings indicate that alternative markets host a large number of ad-aggressive apps, a non-negligible amount of malware, and some markets even allow authors to publish known malicious apps without prompt action.

Motivated by these findings, we present AndRadar, a framework for discovering multiple instances of a malicious Android application in a set of alternative application markets. AndRadar scans a set of markets in parallel to discover similar applications. Each lookup takes no more than a few seconds, regardless of the size of the marketplace. Moreover, it is modular, and new markets can be transparently added once the search and download URLs are known.

Using AndRadar we are able to achieve three goals. First, we can discover malicious applications in alternative markets, second, we can expose app distribution strategies used by malware developers, and third, we can monitor how different markets react to new malware. During a three-month evaluation period, AndRadar tracked over 20,000 apps and recorded more than 1,500 app deletions in 16 markets. Nearly 8% of those deletions were related to apps that were hopping from market to market. The most established markets were able to react and delete new malware within tens of days from the malicious app publication date while other markets did not react at all.

Keywords: Android, App Markets, Measurements, Malware Tracking

1 Introduction

Due to its popularity with nearly 80% market share [15] and open model, Android has become the mobile platform most targeted by cyber criminals. In spite of a small infection rate [16, 17] of devices with mobile malware in the wild, the remarkable increase in the number of malicious applications shows that cyber criminals are actually investing time and effort as they perceive financial gain. Indeed, the typical malicious application includes Trojan-like functionalities to steal sensitive information (e.g., online banking credentials), or dialer-like functionalities to call or text premium numbers from which the authors are paid a commission. The degree of sophistication of Android malware is rather low, although samples of current malware families found in the wild include command-and-control functionalities and attempt to evade detection with in-app downloads of the malicious payload after the installation of a legitimate-looking application. Cyber criminals are focusing more on widespread distribution and naïve signature evasion [23, 32] rather than attack vector sophistication.

Seminal work by Zhou and Jiang [35] reported the existence of 49 distinct malware families according to data collected between 2010 and 2012. Current estimations vary widely, with McAfee reporting about 68k distinct malicious Android applications [19] and Trend Micro counting up to 718k distinct Android "threats" [27] in Q2 2013. However, security vendors and researchers agree that there is an increasing trend of malicious Android apps spotted in the wild, which indicates that the criminals consider this a source for profits. This phenomenon created a business opportunity for new security companies, which according to Maggi et al. [18], created about a hundred anti-malware applications for Android. Interestingly, about 70% of such companies are new players in the antivirus (AV) market.

As with traditional malware, the research community has been focusing on analyzing suspicious programs to identify whether they are malicious or not. In the case of Android, this requires analyzing the application package file (APK), a compressed archive that contains resources (e.g., media files, manifest) and code, including Dalvik executables or libraries, or native code (e.g., ARM or x86). Dynamic, static and hybrid program analysis approaches have also been ported to Android. There is, however, a key difference between traditional malware and Android malware. As we will discuss in Section 2, Android malware is distributed through application marketplaces, which means that there is a wealth of metadata associated with each sample, in addition to the resources contained in each APK. Additional contextual information comes from the infection mechanism, bait-an-switch, which uses an actual benign application distributed through alternative marketplaces to attract victims.

Efficiency is a key requirement for monitoring malware campaigns in the large Android ecosystem. However, we observe that meta information has not been fully leveraged to this end. Indeed, as analyzed in Section 5, past works revolve around features extracted from APK, which in turn implies that the sample is downloaded and processed using static and dynamic analysis techniques, which is time and space consuming.

Motivated by the need for tracking the *distribution* of Android malware across markets, we follow a different approach and propose an alternative way to *identify* them. We demonstrate that the *combination* of lightweight identifiers such as the package name, the developer's certificate fingerprint, and method signatures, creates a very strong identifier, which allows us to track applications across markets. We implemented our approach by building AndRadar, which uses a flexible workflow. First, it applies lightweight fingerprinting to quickly determine if a known sample has been found in a particular market. AndRadar postpones computationally expensive tasks such as binary similarity calculation, so that they can be lazily executed. This allows AndRadar to scan a full market for malware in real-time. Using AndRadar we can infer useful insights about malicious app distribution strategies and the lifetime of malware across multiple markets. For example, for a total of 20,000 crawled apps AndRadar recorded more than 1,500 deletions across 16 markets in a period of three months. Nearly 8% of those deletions were related to apps that were *hopping* from market to market, meaning the authors republished their applications in one or more different markets after they were already deleted from another market. Some markets reacted and deleted new malware within tens of days from the publication date, whereas other markets did not react at all. Interestingly, we were able to measure that the community reacts fast, flagging applications as malicious faster than the market moderation in some cases.

In summary, we make the following contributions:

- We conducted an in-depth measurement on 8 alternative Android marketplaces. In contrast to previous work, we collected the entire set of applications (318,515 overall), and not simply a random subset drawn from each market. With this dataset, we provide preliminary insights on the role of these alternative markets, with a focus on malicious or otherwise unwanted applications.
- We expand our set of observed markets and present AndRadar, a framework for searching a set of markets, in real-time, in order to discover applications similar to a seed of malicious applications. Using a set of distinctive fingerprints that are robust to commonly used repackaging and signature-evasion techniques, AndRadar can scan markets in parallel, and only needs a few seconds to discover a given Android application in tens of alternative application markets.
- Using AndRadar we study and expose the publishing patterns followed by authors of malicious applications on 16 markets. Moreover, our evaluation shows that AndRadar makes harvesting marketplaces for known malicious or unwanted applications fast and convenient.

2 Market Characterization

As we detail in Section 5, previous research shows that in 2011 the majority of malicious or otherwise unwanted Android applications were distributed through so-called *alternative marketplaces*. An alternative marketplace is any web service whose primary purpose is to distribute Android applications. For instance, blogs or review sites that occasionally distribute applications do not qualify as marketplaces. According to our definition, we were able to find 89[4] markets as of June 2013. The *raison d'etre* of such alternative markets depends on three main factors: *country gaps* (i.e., the Google Play Store is inaccessible from certain countries), *promotion* (i.e.,

[4] Although previous work reported 194 markets in 2011 [29], no details such as the URL or name were mentioned.

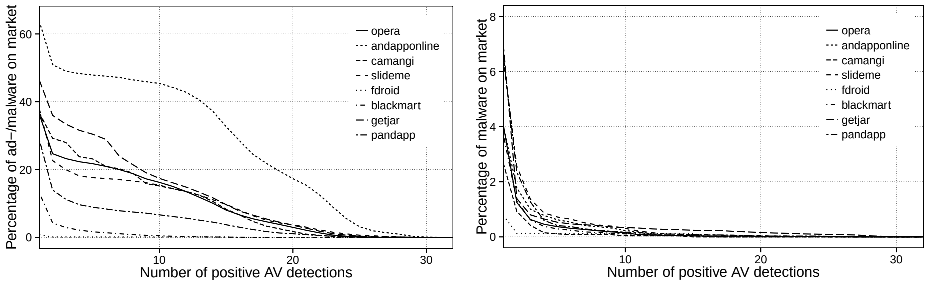


Figure 1: Percentage of applications on alternative markets classified as positives by [1-32] AVs, including adware (left) and excluding adware (right).

markets tailored to help users find new interesting applications), and *specific needs* (i.e., markets that publish applications would be bounced by the Google Play Store).

Regarding malware distribution, since the first measurements conducted in 2011 a lot has changed: Researchers, security vendors and media continuously raise concerns about the explosive growth of Android malware. According to a recent estimate [25], as of 2013 companies have invested about $9 billion in mobile device and network security and installation of anti-malware software has become the de-facto requirement for mobile devices.

2.1 The Role of Alternative Marketplaces

Given the above premises, we wanted to investigate whether alternative marketplaces employ any security countermeasure to avoid the spread of malicious applications. To this end, we conducted a series of probing experiments, in July 2013, aimed at assessing the response of these markets to dangerous applications. We submitted known malicious applications taken from the Android Malware Genome Project [35] to 7 markets (i.e., andapponline, androidpit, appzoom, brothersoft, camangi, opera, slideme) and analyzed their reaction. To deter users from downloading the apps, we included explicit indications that they were malicious and should not be installed. To the best of our knowledge (i.e., by tracking the download counts), those apps were not downloaded. However, certain markets such as andapponline never bounced/removed samples from 10 known families (e.g., DroidKungFu, BaseBridge). This motivated us to conduct a more thorough analysis. Therefore, we crawled 8 alternative marketplaces between July and November 2013 entirely, obtaining 318,515 APKs along with their metadata, which varies across markets (e.g., application name, version, uploader's nickname, category, price, download count, declared permissions). We then extended this crawling experiment, including metadata from a larger set of markets, as described in Section 4.

2.2 Preliminary Findings

Using this initial collection of applications, we set out to answer a the following questions.

Do alternative markets distribute known, unwanted applications? We used VirusTotal to analyze our entire dataset. As illustrated in Figure 1, our analysis showed that the infection rate is not negligible. Even if we exclude adware, there are still about 5–8% malicious applications overall on the crawled markets

Label	#
Android/Generic	2,397
Trojan/AndroidOS.eee	2,119
Trojan.AndroidOS.Generic.A	1,020
AndroidOS/Denofow.B	768
AndroidOS/Denofow.B	765
Suspect.Package.RLO	682
WS.Reputation.1	593
UnclassifiedMalware	555
Android/DrdLight.D!tr	517
AndroidOS/FakeFlash.C	455
Android-PUP/Hamob	443
AndroidOS/FakeFlash.C	428
Application:Android/FakeApp.C	358
Trojan:Android/Downloader.F	339
Andr.Trojan.Zitmo-2	223
Android/DDLight.D!tr	204
Trojan.AndroidOS.FakeFlash.a (v)	192
Android Airpush	182
AndroidOS/FakeFlash.A	174

Table 1: Top malware families found overall.

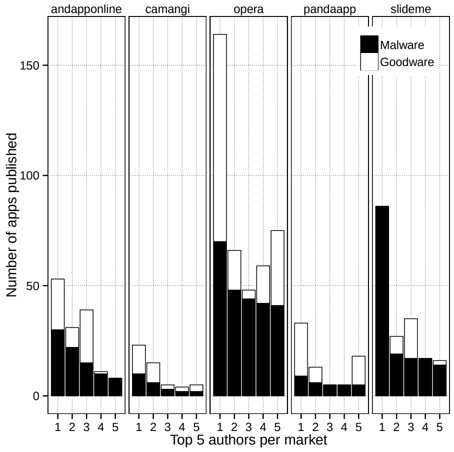


Figure 2: Top 5 authors ranked by number of applications published.

(15,925–25,481 distinct applications detected by at least 10 AVs). This is clearly an underestimation. Interestingly, some markets are specializing in distributing adware. This finding is inline with Symantec's recent report [28], which mentions the "madware" phenomenon, the practice of creating ad-aggressive mobile applications to obtain revenue.

We conducted the remaining preliminary experiments on the applications marked as malicious, excluding adware. We list the ranking of the top families found in Table 1.

Do alternative markets allow the publication of malicious applications? Based on the number of applications published, we ranked the authors of those 5 markets that reported author information reliably (e.g., blackmart simply caches that information from Google Play Store). Unfortunately, as shown in Figure 2, these markets permit the top authors to freely to publish both malicious and benign applications. This finding further amplifies the previous results, because top authors are supposedly well visible and known to the market's operators and community due to the larger number of applications published with respect to other authors.

Do malicious applications have distinctive metadata? Previous work focused on devising static and dynamic features, extracted through program analysis techniques (see Section 5) applied to the APK files, that characterize malicious applications. However, given the central role of alternative markets in malware distribution, we wanted to understand if malware can be identified solely by its *metadata*, meaning all ancillary data available on each market (file size, download count, etc.). As Figure 3 shows, due to repackaging, the file size is a feature to consider: Statistically speaking, malware samples are slightly larger than goodware samples because of the additional malicious code. Similarly, we observed that for those markets that report the download count (e.g., getjar), malware are more downloaded than goodware by at least an order of magnitude. One possible expla-

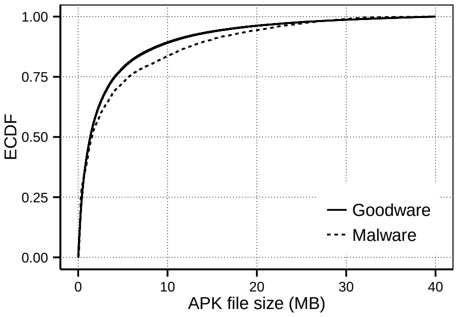


Figure 3: APK file size comparison of malware and goodware: malicious apps are slightly larger than benign apps due to repackaging.

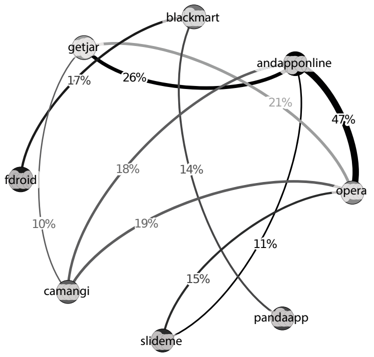


Figure 4: Intersection between markets by MD5: percentage and thickness of edges indicate the percentage of apps in common.

nation for this finding is that malware authors reportedly use app rank boosting services to increase download numbers and thus improve their app's ranking [13].

How are markets related to each other? We calculated the set intersection of APKs across markets by taking the package name or the MD5 hash as the identifier. Due to space constraints we only present the results for the latter in Figure 4. although they both exhibit the same pattern. We can immediately see that the number of shared apps across markets is non-negligible, with some notable examples such as andapponline–opera sharing 47%/59% of MD5s/package names, or andapponline–getjar sharing 26%/38% respectively.

Conclusion. From this preliminary analysis, it appears that alternative markets are not proactively removing malicious applications from their databases. Understandably, the volume of applications to be screened is large and the current analysis methods rely on running expensive and error-prone analyses on each submitted APK. Moreover, given the non-negligible flow of applications across markets, we are concerned that malicious developers may be able to implement a "failover" strategy to have their samples migrate from market to market in order to hinder removal.

These findings motivate us to devise an Android market radar, called AndRadar. AndRadar uses lightweight and transparent techniques that permit the quick scanning of alternative markets for malicious or otherwise unwanted applications and allow us to track apps and their metadata across different markets.

3 Android Market Radar (AndRadar)

In this section we present the architecture of AndRadar. First, we discuss various challenges we faced while designing and implementing AndRadar, and then we describe its various components in detail.

3.1 Challenges

AndRadar aims at discovering a particular Android application, possibly indicated as malware or otherwise unwanted applications by an AV scanner, in the official Google Play Store as well as alternative markets. This is a non-trivial task as we show in this part. Below we list the most significant challenges we had to overcome, while building the prototype.

Marketplaces Plethora. During our preliminary experiments discussed in Section 2, we found 89 alternative marketplaces, run by companies or individuals, whose quality in terms of security aspects is questionable. As demonstrated by our marketplace study, which took months to complete, crawling markets is challenging. First, because space and time requirements increase quickly with the number of markets. Secondly, and most importantly, each market runs its own software. This essentially means that for each market we want to monitor we need to analyze its API for searching and downloading apps. Normally, this involves discovering two URLs, one for searching for an application and one for downloading a discovered application along with its metadata. Unfortunately, for many markets this process is not straightforward. For example, many of them strictly require user authentication—especially markets with specialized content, like adult content—or are provided in the form of a mobile application, which needs manual reverse-engineering for revealing the market API. Finally, while running AndRadar we also experienced cases where markets, like for example Google Play and appchina, changed their web templates during our experiments. Changes in a market's web templates essentially requires for us to carry out further adjustments in the engine we use for extracting application metadata.

Application Mutation. The diversity of the marketplaces is not the only challenge we have to overcome. Applications can slightly mutate from market to market. This can be done due to legitimate reasons, for example two markets host two different versions of a particular application, or because they are repackaged by another author either to add additional functionality missing from the original application, or to profit from a popular application by including advertising libraries or malicious code [29]. Detecting repackaged applications, maybe the most popular form of Android malware, has been the target of recent related work [7,33,34]. AndRadar's primal goal is not to detect if a particular application has been repackaged, but locating an application – possibly malware – across different marketplaces. Research in repackaged-application detection is orthogonal to AndRadar. Nevertheless, it can substantially assist AndRadar in discovering repackaged versions of applications across different alternative markets. Recall that the common wisdom suggests that popular apps hosted in the official market are enhanced with malicious functionality, repackaged and published to alternative marketplaces. We envision that, due to the immediate popularity gained by alternative markets and to the continuously growing defense systems in the official Google Play Store, malware authors will further target alternative markets. Therefore we expect them to start repackaging legitimate apps found in popular alternative markets and then publishing the produced malware in less popular markets. In such cases, AndRadar can use existing algorithms and heuristics for real-time detection of repackaged applications across multiple marketplaces.

3.2 Architecture Overview

We now present an overview of AndRadar's architecture. In a nutshell, AndRadar's task is to probe a number of marketplaces for malware and, if found, track it. Figure 5 shows how the components of AndRadar interact to achieve this task.

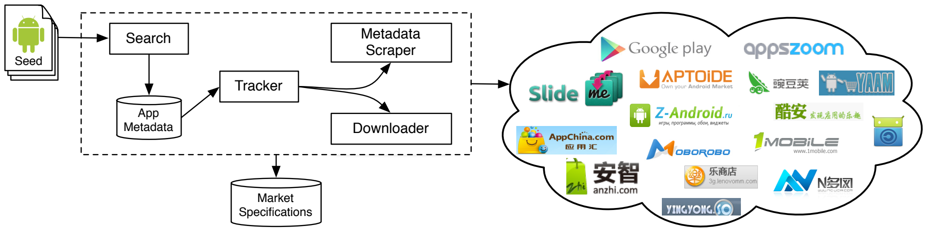


Figure 5: Overview of AndRadar's architecture: The *seed*, which is composed by apps that have been flagged as malware, is used as input to the *search* component for locating apps across markets. Once an app is found, the *tracker* downloads and stores additional metadata.

Essentially, AndRadar has three core components: The first one is the *seed*, which is composed by a set of apps that have been flagged as malware by a set of tools or services. This is the input set that AndRadar uses for locating apps across alternative markets. The second component is the *search* component. For each app in the seed, AndRadar uses a set of crawlers for discovering the app in an alternative market. Finally, the third component is the *tracker*, which, once an app is found, downloads its additional metadata and keeps it in storage for further statistics. We now look into each of the three components in detail.

3.3 Seed Sources and Content

To begin with, AndRadar requires a set of known malware or otherwise unwanted applications that we call the *seed*. Because of its dynamic, online functionality, AndRadar works best with a continuous, accumulating feed of malicious apps in contrast to a static set. Apps for the seed can come from a variety of sources including new additions to manually vetted malware repositories, feeds from submissions to AV scanning services that are detected by multiple scanners as malicious, or submissions to dynamic analysis sandboxes.

For our prototype AndRadar receives feeds from VirusShare [2], submissions to VirusTotal [3] that trigger > 10 AV signatures, and submissions that Andrubis [1,30] flagged as suspicious during dynamic analysis. However, AndRadar could be easily extended to add further sources for malware such as submissions to AndroTotal [18].

Each app in the seed is characterized by four identifiers that allow us to match two apps at different levels of confidence (see Table 2 for a summary):

Package name. The package name is the "official" identifier of an app. It serves as an installation-time ID, i.e., no two apps on a given device can share the same package name. Some markets, such as Google Play, use it also as a unique reference, but in principle developers are not restricted from creating an app with an already existing package name. Therefore, in the context of AndRadar which operates on a multi-market domain, we use the package name to locate apps inside a market (see Section 3.4) and treat it as a *weak match* between two apps. However, AndRadar is not restricted to this identifier as we further will discuss in Section 6.

Fingerprint. Apps in Android are signed with the private key of their developer. Android uses this certificate to enforce update integrity by only allowing updates signed with the same key, as well as sharing of resources and permission inheritance between apps from the same author [4]. We can thus use the fingerprint of the certificate used to sign the app as a further identifier. Since the key is specific to an author, a match of the fingerprint is a strong indicator that the matching apps stem from the same author, unless the author has shared her private key or the author is using the key pair that is publicly available with the Android source code. We thus treat a match of package name and author fingerprint as a *strong match*.

Method signatures. By leveraging Androguard [8,22] we can generate signatures of the methods in the application code. A signature is an abstract model of a method's intraprocedural control flow, enriched with information on the package of further called methods. To compare signatures, Androguard uses the normalized compression distance. For AndRadar, we limit the scope of the signatures to methods that are either in the main package or in the package that contains the app's main activity, thus excluding third-party libraries that and would skew the comparison results. We define everything above a 90% code similarity to be a *strong match*. In addition, we define the combination of a method signature, fingerprint and package name match as a *very strong match*.

MD5 hash. In a very straightforward way, a match between the MD5 hash of two APK files means that two applications are identical, i.e. a *perfect match*.

App identifier	**Match level**
MD5	perfect match
Package name, fingerprint, method signatures	very strong match
Package name, method signatures	strong match
Package name, fingerprint	strong match
Package name	weak match

Table 2: Different match levels based on app identifiers.

3.4 Search

The *search* component probes markets for a given app, based on its package name. We chose the package name for our searching procedure, since it provides a strong heuristic in order to identify a sample based in a seed entry inside a market and some markets use it to uniquely identify apps in their app catalog. Of course, as we discuss in Section 6, a malware author could randomize the package name from market to market, but this would actually run against the malware author's own scheme when trying to trick users into downloading his repackaged version of a popular application. Thus essentially, a malicious app trying to remain hidden from AndRadar would substantially reduce its visibility to potential victims. As a future extension, we may add options to search for words appearing in the title, or through other metadata that users might use to locate an application inside a market. This, however, would require AndRadar to track and download multiple candidate apps and their metadata from each market in order to locate samples matching the seed application.

For markets such as Google Play, appchina, anzhi, wandoujia or coolapk, that use the package name as an internal reference to the apps the lookup is straightforward, as the package name is typically part of the app's URL in the market. Other markets use different internal identifiers and thus require a more elaborate search procedure. In that case we split the package name along the separators and feed the individual parts to the market's search interface, discarding well-known common parts such as, e.g., "com". Once the package name is located on the results page, the search is considered finished. Otherwise, we continue by crawling the individual market listings that are returned by the search query.

Finally, based on an author's publishing habit, apps might appear in our seed and later be released to a new market at a later point in time. As a consequence the search component probes all the markets for all malicious apps at regular intervals regardless whether they have been located before or not.

3.5 Tracking

Once the search component finds an app in a market, the corresponding market listing will be investigated by the *tracker*. The tracker first invokes the *downloader* to fetch the sample from the market. The downloaded app is matched with the sample in the seed using the set of similarity features summarized in Table 2. In Figure 6 we present the flow chart of AndRadar's matching algorithm. The tracker uses the *scraper* to obtain market-based metadata of each sample, from each monitored market at regular intervals. Metadata includes the reported version of an app as well as its popularity metrics such as download count, price, user ratings and reviews, update date, delete date, etc. If an app's metadata information has changed, indicating a possible update, the new version of the app is downloaded and kept in storage.

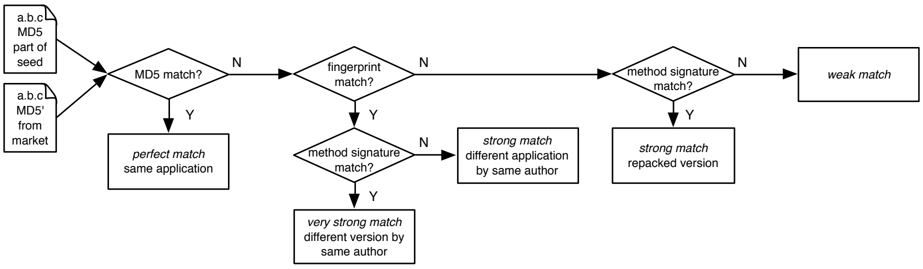


Figure 6: Flow chart of AndRadar's application matching.

4 Evaluation and Case Study

In this section we evaluate AndRadar in terms of performance, and use the system to reveal insights about the behavior of particular applications, characterized as possibly malware, across multiple markets.

4.1 Performance

AndRadar tracks apps in multiple markets in a parallel fashion. For the purposes of the study presented in this paper, we have incorporated 16 different markets. The time needed to search and download a particular app across all the markets is illustrated in Figure 7. Naturally, downloading is slower than searching, but both operations take just a few seconds to complete for the majority of markets. However, the download of an app is only initiated when the metadata information indicates a possible update. Furthermore, both operations depend on the network conditions, as well as the load the market is experiencing at the time, but since

AndRadar crawls all markets in parallel, we are only constrained by the slowest market. We list the amount of apps we can track in each market per day in Table 3. As it can be seen we are able to track tens of thousands of apps daily.

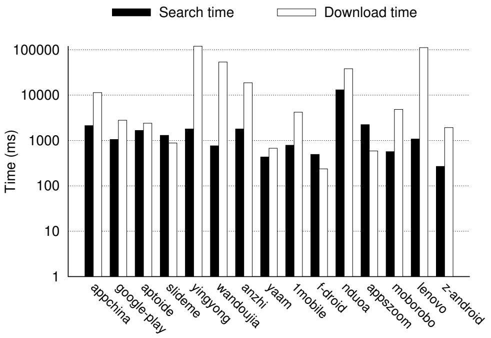


Figure 7: Average time needed for searching and downloading an app on each market. Since AndRadar handles all markets in parallel, searching and downloading a particular app on *all* markets is constrained by just the slower market.

Market	S	D	Total	#/day
f-droid	0.49s	0.24s	0.73s	118163
yaam	0.43s	0.67s	1.11s	77996
slideme	1.30s	0.88s	2.18s	39662
z-android	0.27s	1.93s	2.20s	39319
appszoom	2.23s	0.59s	2.82s	30605
google-play	1.06s	2.79s	3.85s	22441
aptoide	1.67s	2.41s	4.08s	21199
1mobile	0.79s	4.20s	4.99s	17305
moborobo	0.57s	4.83s	5.40s	15994
appchina	2.13s	11.36s	13.49s	6406
anzhi	1.80s	18.69s	20.49s	4217
nduoa	13.06s	38.18s	51.25s	1685
wandoujia	0.76s	53.65s	54.41s	1587
lenovo	1.08s	111.43s	112.51s	767
yingyong	1.80s	119.56s	121.35s	711

Table 3: Average time needed for searching (S) and downloading (D) an app on each market and number (#) of apps we can track in each market per day.

4.2 Case Study

AndRadar essentially gives us the opportunity to collect data about an app in multiple markets, study the multi-market behavior of the app, and, possibly, identify publishing patterns followed by app developers. For instance, if we use AndRadar with a sample of (possibly) malicious apps, we can understand better how malicious apps behave across different markets. In this section, we present the insights we obtained by crawling 20,000 apps in a daily manner between August and December 2013 in 16 markets. These apps matched applications in our seed at least by package name and were identified according to the process described in Section 3.

For the purpose of this case study we split the sample of tracked apps in two sets: a) *deleted*, a set that contains all apps that have been deleted at least once from a market during our observation period, and b) *non-deleted*, a set of apps that have never been deleted from any of the markets.

Since AndRadar checks each app located in one of the markets against the malicious app from the original seed using a set of similarity features (detailed in Table 2 and Figure 6), we have a spectrum of confidence regarding the maliciousness of the collected apps. In Figure 8 we plot the distribution of the collected dataset (across both deleted and non deleted apps) against the similarity features used.

If we identify an app with a perfect match (MD5) that is removed after a period of time (corresponding to the black bar in the deleted group in Figure 8), we assume that the market administrators did this for a reason and found something something malicious bout the app, thus strengthening our initial suspicion. Conversely, on

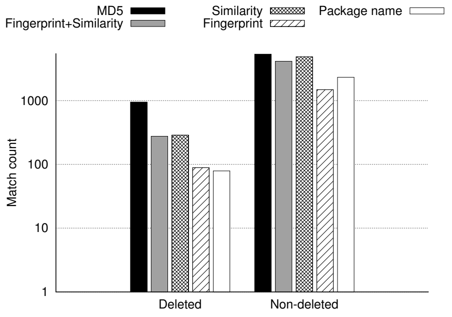


Figure 8: Number of deleted and non-deleted apps per matching type across all markets.

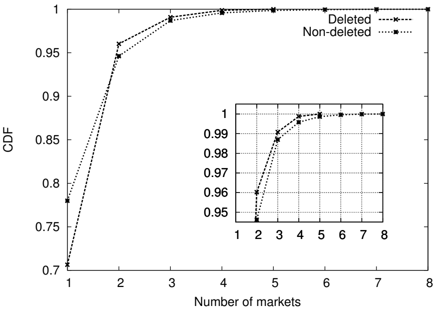


Figure 9: Deleted and non-deleted applications located in multiple markets at the same point in time.

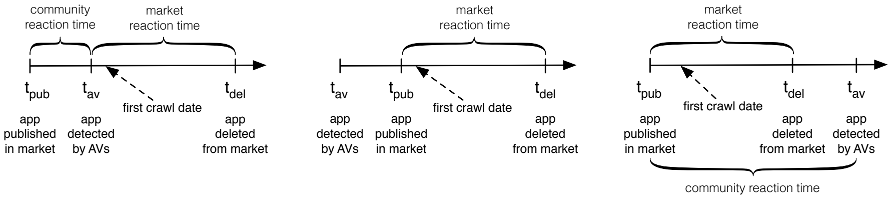
a weak package name match, a missing reaction from the market administrators (corresponding to the white bar in the non-deleted group in Figure 8) indicates that the app located in the market is the benign version the author of the malicious seed app used as a disguise.

In Figure 9 we plot the CDF of the two sets, deleted and non-deleted apps, over the number of markets each app has been located on by AndRadar at the same point in time. This figure justifies our initial concern that malware authors indeed leverage the plethora of app markets to distribute malware. A non-negligible portion of apps simultaneously leverages more than five markets for distribution (roughly 1/3 of the markets we have been monitoring). As an example, we were able to locate the malicious app *King Pirate* (`com.letang.game101.en.f`) in five different markets. In some of those markets the app has been available for over a year and thus reached a considerable amount of downloads. To date it was only deleted from one of the markets we monitored:

1. *appchina:* online since 15-03-2012, 1,000-5,000 downloads
2. *aptoide:* online since 25-05-2012, 270 downloads
3. *wandoujia:* online since 12-08-2012, 1,430 downloads
4. *1mobile:* online since 26-07-2013, 25,808 downloads
5. *lenovo:* online from 17-10-2012 to 18-10-2013

The app advertises itself as a legitimate game available in the Google Play Store[5]. The repackaged version adds the functionality to manipulate SMS, install additional packages, and perform payments to the original game. It was first submitted to VirusTotal in September 2012, flagged by the first AV scanners in December 2012, and since then identified by 16 scanners as a Trojan horse, under the names Android/Ksapp.D or Android/Qdplugin.A. Clearly, in cases like this, it is desirable for market operators to remove the application from their catalog as soon as possible. To aid them in doing so, we are going to integrate an automated notification system into AndRadar.

[5] `https://play.google.com/store/apps/details?id=com.letang.kpe`



(a) *Normal Lifecycle:* An app is deleted from a market after it has been flagged by AVs.

(b) *Malware Hopping:* An app is published to a market after it already has been flagged by AVs.

(c) *Market Self-Defense:* An app is deleted from a market before it has been flagged by AVs.

Figure 10: Patterns for the lifecycle of a malicious app in a market.

Type	Number	Percentage
Normal	1,508	90.57%
Possibly Malware Hopping	131	7.86%
Possibly Market Self-Defense	26	1.56%

Table 4: Distribution of the lifecycle patterns presented in Figure 10 for all deleted apps: The large majority of apps follows the "Normal" case, but we also found evidence of market hopping and market self-defense.

Market	Deleted Apps
google-play	1281
appchina	236
anzhi	83
wandoujia	48
lenovo	15
1mobile	1
aptoide	1

Table 5: Distribution of deleted apps across markets.

Finally, we take a look at how fast both the security community and the application markets react to new malware and whether a multi-market strategy enhances the lifetime of malware. We identified three typical patterns for the lifecycle of a malicious app:

Normal. In the most common case, an app is first published in a market at t_{pub}, it is later identified by the community and flagged by (some) AVs at t_{av}, and at a later point deleted from the market at t_{del}. We define as the *community reaction time* the period $t_{av} - t_{pub}$ and as *market reaction time* the period of $t_{del} - t_{av}$. We depict this behavior in Figure 10 (a).

Malware Hopping. In this scenario, malicious apps are *republished* in different markets after they have been flagged as malware by AVs. In this pattern, an app is published in a market at t_{pub}, but has been identified by the community at an earlier point t_{av}. At a later point the app is deleted from the market at t_{del}. We define the period of $t_{del} - t_{pub}$ as the market reaction time. We depict this behavior in Figure 10 (b).

Market Self-Defense. Markets can sometimes filter malicious apps even before they are flagged by AVs. In some instances, an app is published in a market at t_{pub}, at a later point the app is deleted from the market at t_{del}, and at even a later point the app is flagged as malware by AVs. Again, $t_{del} - t_{pub}$ is the market reaction time. We depict this behavior in Figure 10 (c).

We present the distribution of all deleted apps among these three scenarios in Table 4. The majority of apps follow the "Normal" case, but AndRadar could also

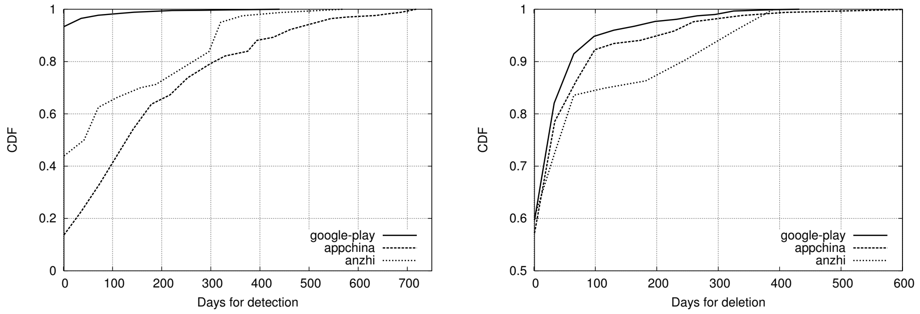


Figure 11: Time needed for AVs to detect apps as malware (*community reaction time*, left) and time needed for markets to delete apps after they have been flagged as malware (*market reaction time*, right).

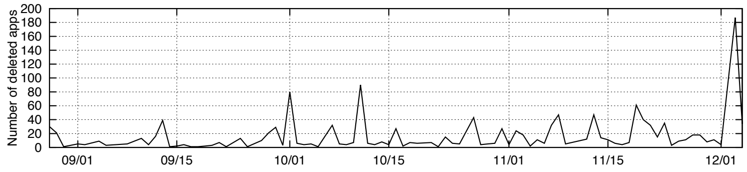


Figure 12: Number of apps deleted from Google Play on a daily basis between September and December 2013.

identify apps that followed the other two cases, finding evidence that malicious apps jump from market to market, possibly for survival, and also evidence that some markets remove apps using some internal security mechanism.

For all deleted apps that follow case (a) in Figure 10 we measured the community reaction time, which is the time needed for AVs to flag a particular app, once this app was published in a market, and the market reaction time, which is the time needed for a market to delete an app that was flagged by an AV as malware. We present the distribution of app deletions per market in Table 5 and we depict the results for the three markets that deleted the most applications in Figure 11. The following insights can be gained from this figure:

First, each market has a different reaction behavior. It is evident that apps that are published in Google Play reach the AVs community faster than those in other markets. The majority of Google Play apps are submitted to AVs just a few days after publication.

Second, Google Play is also the fastest market to react when apps are flagged as malicious by AVs. It takes tens of days for Google to delete the malicious apps. The other two markets (appchina and anzhi) have a similar, but slower, behavior.

Third, there is a small but not negligible fraction (less than 4%) of apps which are deleted from markets only after several months (in some cases after more than a year). After manual inspection of these incidents, we discovered that such malicious apps fall into the *gray area* of adware, and are thus sometimes considered not dangerous enough to be removed. For example, due to policy changes Google only recently decided to remove apps including intrusive ad libraries such as AirPush from the Play Store [24]. In another recent example, researchers discovered "vulnaggressive" (aggressive and vulnerable) versions of the ad library AppLovin being used in popular apps, that were subsequently updated or removed [31]. As illustrated in Figure 12, developments like this can be recorded by AndRadar. Google seems to clean its store in regular intervals, with the number of deletions increasing after the market policy changes came into effect at the end of September and the vulnerabilities in AppLovin were disclosed. In fact, out of the 1,749 apps for which we recorded deletion events on Google Play between August 28 and December 4 2013, 1,517 apps are detected at least by one AV scanner as adware. Almost 90% of those apps include libraries such as AirPush, Leadbolt, AdWo and Apperhand, that display push notification ads [26] now being banned by Google's new policy. Some of those applications were in the market for up for more than a year and were downloaded between 100,000 and 500,000 times. For example, the application with package name `com.airbit.soft.siii.oceano` was deleted from Google Play after 409 days of its upload date and it is flagged by many AV vendors as an AirPush adware.

5 Related Work

Android security has been covered extensively in the literature [9] and is still a major research topic. Furthermore, many generic measurements of mobile application marketplaces have been conducted such as a recent study by Petsas et al. [21], but we will focus on the ones related to malware.

The practice of repackaging applications was studied in DroidMOSS [34], where the authors propose a fuzzy hashing similarity metric to compare two APKs and determine whether one is the repackaged version of the other. With their tool, the authors determined in March 2011 that 5–13% of applications found on 6 alternative marketplaces (slideme, freewarelovers, eoemarket, goapk, softportal, proandroid) contain repackaged versions of applications obtained from the Google Play Store.

The approach proposed in Juxtapp [14] determines whether applications contain instances of known, flawed code, exhibit code reuse that indicates plagiarism, piracy, or are (repackaged) variants of known malware. Differently from DroidMOSS [34], this approach does not explicitly concentrate on repackaging (although it effectively finds repackaged applications), thus it is more generic. Moreover, it has a strong focus on scalability, proposing a similarity metric that is applicable to map-reduce frameworks. They show that 100 minutes of computation on 100 8-core machines with 64GB of RAM are sufficient to analyze 95,000 distinct APKs. Unfortunately, obtaining the APKs is the bottleneck, as we showed in Section 4.1.

Vidas et al. [29] conducted a large-scale measurement on 194 alternative Android markets (of which a list was not disclosed, to the best of our knowledge) in October 2011, collecting 41,057 applications. Their key finding was that certain markets almost exclusively distribute repackaged applications containing malware. They propose to counteract the spread of repackaged applications by re-designing how markets authenticate submitted applications. All three approaches [14, 29, 34] require downloading the APKs, and processing their manifest and code offline. As a result, for instance in the study by Vidas et al. [29], which is by far the most extensive of the three, the numbers suggest that the authors have sampled only an average of 211 applications per market, that is, very few compared to the overall market sizes. With our lightweight market monitoring technique we can monitor even the biggest alternative markets such as lenovo, containing around 400,000 applications, or the official Google Play Store with around 800,000 applications [20].

The authors of DroidRanger [36] proposed a permission-based and bytecode-based fingerprinting approach to tell malicious and benign applications apart. With this approach they conducted a measurement on 5 markets (including the Google Play Store) in May 2011, analyzed 204,040 applications, and determined that 211 applications were exhibiting a malicious pattern. In the same fashion, RiskRanker [12] tries to identify certain behaviors – observed in malware – in a given app and associate a risk with it. Both of these works focus on finding or inferring malware on markets; we took a step further, proposing an approach that is fast enough to allow *tracking* malware across markets over time.

Building on the aforementioned findings, Zhou et al. [33] propose an approach to decouple primary from non-primary application modules. The authors observe that the malicious payload, which is piggybacked to legitimate applications, simply adds non-primary modules. Based on this, they propose a feature vector able to tell repackaged applications and their respective legitimate applications apart. They applied their technique to 84,767 applications collected from 7 markets (slideme, freewarelovers, eoemarket, goapk, softportal, proandroid, Google Play Store) in March 2011, and reported that the practice of repackaging apps ranges between 0.97 and 2.7%.

MAST [5] has a goal similar to ours: Finding fast analysis techniques that scale to match the extensiveness of today's markets. MAST is trained on a small set of benign and malicious applications, from which features such as permissions, intents, or native code information are extracted. Then, it uses multiple correspondence analysis (MCA) to triage new applications.

Quantifying the similarity between two Android applications is currently an active research topic. Ready-to-use tools such as Androsim [22], part of Androguard project [8], can assist reverse engineers, but exhibit accuracy and scalability issues. Proposed almost concurrently with Juxtapp [14], DNADroid [7] leverages information from the dependency graph to create a structural comparison criterion based on graph isomorphism, which allows to find pairs of matching methods to detect plagiarized applications. Although their goal are different from ours, their methods can in principle be applied to track malware across markets.

Another example of applying plagiarism detection is described in AdRob [6,11], where the authors concentrate on the problem of ad-aggressive applications. Indeed, repackaging (paid) applications to incorporate ad libraries and distribute the resulting applications on alternative markets seems to be a profitable, illicit business. The authors' estimations were based on monitoring the HTTP advertising traffic generated by 265,359 applications obtained from 17 alternative markets. As ad-based revenue models are not considered malware, this work is orthogonal to ours. Indeed, in our preliminary market characterization described in Section 2 we explicitly removed adware samples. Moreover, their work depends on a static and dynamic analysis phase, which is more expensive than our lightweight, metadata-based approach.

The main difference of related work with AndRadar, is that other approaches all focus on crawling (a subset of applications on) alternative markets and performing expensive static and dynamic analysis on APK files, in many cases with modified

Android platforms. Contrarily, our system requires just a public interface to query apps in a market, and is therefore much faster, scalable and lightweight.

6 Limitations and Future Work

For our prototype AndRadar was configured to discover apps by their package name as the monitored markets uniquely identify applications by this identifier. Also, previous work reported that that malware authors tend to use valid and legitimate looking package names in an effort not to attract attention [29, 35]. A recent report by F-Secure [10] found 23% of malicious apps posing as legitimates ones by imitating their package name. Consequently, they classified apps using the original package and application name but requesting additional permissions compared to the original application as malicious. Alternatively, in order to counter malicious app authors randomizing the package name or simply modifying single letters similar typosquatting, AndRadar can query markets for other identifiers. Possible candidates are application titles, parts of their description or image characteristics of the screenshots advertising the apps functionality. In order to attract users and entice them into downloading their apps, malicious authors need an identifiable "brand", e.g. by piggybacking on popular apps from the official market. Thus, if malicious authors decide to evade the discovery of their apps by AndRadar, this would invariably lower their visibility to users.

Current binary similarity measurements for Android exhibit accuracy and scalability issues. AndRadar tries to mitigate this limiting the scope of the comparison to the main application's code, and by lazily executing such computationally expensive tasks. However, due to its flexible architecture, AndRadar can be extended to use more scalable binary comparison techniques and also include other characteristics from apps' resources or their visual similarity.

For future work we can incorporate an automatic notification system that warns market operators about the presence of malicious applications in their app catalog. Depending on the type of match of the application found in the market with the application in the original malicious seed, AndRadar can issue warnings with different levels of confidence. Furthermore, we plan to offer the app discovery mechanism of AndRadar through a public interface in order to allow security researchers and developers concerned about plagiarized versions of their apps to search alternative markets in real-time.

Finally, since AndRadar tracks different versions of a malicious applications across markets, as well as updated versions of an application in a single market, we can leverage this data to identify further publishing patterns and the evolution of the malicious functionality over time in future work.

7 Conclusion

Our work started from an in-depth measurement performed on 8 alternative Android marketplaces, by collecting their entire set of applications and analyzing various characteristics. This measurement provided us with significant preliminary insights on the role of these alternative markets, with a focus on malicious or otherwise unwanted applications. This is by far the most up-to-date measurement of the alternative marketplaces. Even the most recent work that we surveyed is based on data collected back in 2011.

Our findings motivated us to design and implement AndRadar, a complete framework to monitor alternative markets for malware in real-time, leveraging the wealth of metadata associated with each sample. We demonstrated that the *combination* of lightweight identifiers such as the package name, the developer's certificate fingerprint, and method signatures, creates a very strong identifier, which allows us to track applications across markets.

Thanks to the efficiency by design of AndRadar, we were able to measure the lifetime of malware across multiple markets in real-time. For example, we tracked more than 1,500 app deletions across 16 markets over a period of three months. We discovered that nearly 8% of the deletions were related to apps that were hopping from market to market.

AndRadar was also able to identify and track malicious apps still available in a number of alternative app markets. For future work we plan to integrate an automated notification system into our framework, that informs market operators about potentially malicious applications in their catalog. We believe that efforts such as ours can be successfully leveraged by official marketplaces to "predict" upcoming spreads, so as to provide early warnings and prompt remediations. Indeed we found out that, for some markets (i.e., Google Play Store), the community contribution is essential to quickly react against published malicious or unwanted apps.

Furthermore, we can also leverage the different versions of malicious apps that AndRadar tracks to identify further publishing patterns as how malware authors change the malicious functionality of their apps over time. This is part of our future work.

Acknowledgments. This work was supported in part by the project ForToo, funded by the Directorate-General for Home Affairs under Grant Agreement No. HOME/2010/ISEC/AG/INT-002 and by the FP7 projects NECOMA, OPTET, SysSec, under Grant Agreements No. 608533, No. 317631 and No. 257007. It was also supported in part by the FP7-PEOPLE-2010-IOF project XHUNTER, No. 273765, MIUR FACE Project No. RBFR13AJFT, and FFG – Austrian Research Promotion under grant COMET K1.

References

1. Anubis: Analyzing Unknown Binaries. `http://anubis.iseclab.org`
2. VirusShare. `http://www.virusshare.com`
3. VirusTotal. `http://www.virustotal.com`
4. Barrera, D., Clark, J., McCarney, D., van Oorschot, P.C.: Understanding and Improving App Installation Security Mechanisms Through Empirical Analysis of Android. In: Proceedings of the 2nd ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM) (2012)
5. Chakradeo, S., Reaves, B., Traynor, P., Enck, W.: MAST: Triage for Market-scale Mobile Malware Analysis. In: Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec) (2013)

6. Chen, H.: Underground Economy of Android Application Plagiarism. In: Proceedings of the 1st International Workshop on Security in Embedded Systems and Smartphones (SESP) (2013)
7. Crussell, J., Gibler, C., Chen, H.: Attack of the Clones: Detecting Cloned Applications on Android Markets. In: Proceedings of the 17th European Symposium on Research in Computer Security (ESORICS) (2012)
8. Desnos, A., Gueguen, G.: Android: From Reversing To Decompilation. In: Black Hat Abu Dhabi (2011)
9. Enck, W., Octeau, D., McDaniel, P., Chaudhuri, S.: A Study of Android Application Security. In: Proceedings of the 20th USENIX Security Symposium (2011)
10. F-Secure: Threat Report H2 2013. http://www.f-secure.com/static/doc/labs_global/Research/Threat_Report_H2_2013.pdf (March 2014)
11. Gibler, C., Stevens, R., Crussell, J., Chen, H., Zang, H., Choi, H.: AdRob: Examining the Landscape and Impact of Android Application Plagiarism. In: Proceedings of 11th International Conference on Mobile Systems, Applications and Services (MobiSys) (2013)
12. Grace, M., Zhou, Y., Zhang, Q., Zou, S., Jiang, X.: RiskRanker: Scalable and Accurate Zero-day Android Malware Detection. In: Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys) (2012)
13. Gu, L.: The Mobile Cybercriminal Underground Market in China. Tech. rep., Trend Micro (March 2014), http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-the-mobile-cybercriminal-underground-market-in-china.pdf
14. Hanna, S., Huang, L., Wu, E., Li, S., Chen, C., Song, D.: Juxtapp: A Scalable System for Detecting Code Reuse Among Android Applications. In: Proceedings of the 9th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA) (2012)
15. IDC: Apple Cedes Market Share in Smartphone Operating System Market as Android Surges and Windows Phone Gains. http://www.idc.com/getdoc.jsp?containerId=prUS24257413 (August 2013)
16. Lever, C., Antonakakis, M., Reaves, B., Traynor, P., Lee, W.: The Core of the Matter: Analyzing Malicious Traffic in Cellular Carriers. In: Proceedings of the 20th Annual Network & Distributed System Security Symposium (NDSS) (2013)
17. Ludwig, A., Davis, E., Larimer, J.: Android - Practical Security From the Ground Up. In: Virus Bulletin Conference (2013)
18. Maggi, F., Valdi, A., Zanero, S.: AndroTotal: A Flexible, Scalable Toolbox and Service for Testing Mobile Malware Detectors. In: Proceedings of the 3rd Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM) (2013)
19. McAfee Labs: McAfee Threats Report: Second Quarter 2013. http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q2-2013.pdf (August 2013)
20. One Platform Foundation: List of Android Appstores. http://www.onepf.org/appstores/
21. Petsas, T., Papadogiannakis, A., Polychronakis, M., Markatos, E.P., Karagiannis, T.: Rise of the Planet of the Apps: A Systematic Study of the Mobile App Ecosystem. In: Proceedings of the 2013 Conference on Internet Measurement Conference (IMC) (2013)
22. Pouik, G0rfi3ld: Similarities for Fun & Profit. Phrack Magazine 14(68) (2012)
23. Rastogi, V., Chen, Y., Jiang, X.: DroidChameleon: Evaluating Android Anti-malware Against Transformation Attacks. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (ASIACCS) (2013)
24. Ruddock, D.: Google Pushes Major Update To Play Developer Content Policy, Kills Notification Bar Ads For Real This Time, And A Lot More.

http://www.androidpolice.com/2013/08/23/teardown-google-pushes-major-update-to-play-developer-content-policy-kills-notification-bar-ads-for-real-this-time-and-a-lot-more/ (September 2013)

25. Signals and Systems Telecom: The Mobile Device & Network Security Bible: 2013–2020. Tech. rep. (September 2013), http://www.reportsnreports.com/reports/267722-the-mobile-device-network-security-bible-2013-2020.html
26. Simon, Z.: Adwares. Are they viruses or not? http://androidmalwareresearch.blogspot.gr/2012/07/adwares-are-they-viruses-or-not.html (July 2012)
27. Trend Micro: TrendLabs 2Q 2013 Security Roundup. http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/reports/rpt-2q-2013-trendlabs-security-roundup.pdf (August 2013)
28. Uscilowski, B.: Mobile Adware and Malware Analysis. Tech. rep., Symantec (October 2013), http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/madware_and_malware_analysis.pdf
29. Vidas, T., Christin, N.: Sweetening Android Lemon Markets: Measuring and Combating Malware in Application Marketplaces. In: Proceedings of the 3rd ACM Conference on Data and Application Security and Privacy (CODASPY) (2013)
30. Weichselbaum, L., Neugschwandtner, M., Lindorfer, M., Fratantonio, Y., van der Veen, V., Platzer, C.: Andrubis: Android Malware Under The Magnifying Glass. Tech. Rep. TR-ISECLAB-0414-001, Vienna University of Technology (2014)
31. Zhang, Y., Xue, H., Wei, T., Song, D.: Monitoring Vulnaggressive Apps on Google Play. http://www.fireeye.com/blog/technical/2013/11/monitoring-vulnaggressive-apps-on-google-play.html (November 2013)
32. Zheng, M., Lee, P., Lui, J.: ADAM: An Automatic and Extensible Platform to Stress Test Android Anti-virus Systems. In: Proceedings of the 10th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA) (2013)
33. Zhou, W., Zhou, Y., Grace, M., Jiang, X., Zou, S.: Fast, Scalable Detection of "Piggybacked" Mobile Applications. In: Proceedings of the 3rd ACM Conference on Data and Application Security and Privacy (CODASPY) (2013)
34. Zhou, W., Zhou, Y., Jiang, X., Ning, P.: Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces. In: Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy (CODASPY) (2012)
35. Zhou, Y., Jiang, X.: Dissecting Android Malware: Characterization and Evolution. In: Proceedings of the 33rd IEEE Symposium on Security and Privacy (2012)
36. Zhou, Y., Wang, Z., Zhou, W., Jiang, X.: Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets. In: Proceedings of the 19th Annual Network & Distributed System Security Symposium (NDSS) (2012)

Appendix

Marketplace	**Website**	**S**	**R**
1mobile	`www.1mobile.com`		✓
andapponline	`www.andapponline.com`	✓	
anzhi	`www.anzhi.com`		✓
appchina	`www.appchina.com`		✓
appszoom	`www.appszoom.com`		✓
aptoide	`www.aptoide.com`		✓
blackmart	`www.blackmart.altervista.org`	✓	
camangi	`www.camangimarket.com`	✓	
coolapk	`www.coolapk.com`		✓
f-droid	`f-droid.org`	✓	✓
getjar	`www.getjar.mobi`	✓	
google-play	`play.google.com`		✓
lenovo	`app.lenovo.com`		✓
moborobo	`store.moborobo.com`		✓
nduoa	`www.nduoa.com`		✓
opera	`apps.opera.com`	✓	
pandaapp	`download.pandaapp.com`	✓	
slideme	`slideme.org`	✓	✓
wandoujia	`www.wandoujia.com`		✓
yaam	`yaam.mobi`		✓
yingyong	`www.yingyong.so`		✓
z-android	`z-android.ru`		✓

Table 6: Marketplaces examined in our market study (S) and monitored by AndRadar (R).