Delaying Informed Consent: An Empirical Investigation of Mobile Apps' Upgrade Decisions

Raveesh Mayya
Leonard N. Stern School of Business
New York University, New York, NY 10012.

Siva Viswanathan
Robert H Smith School of Business
University of Maryland, College Park, MD 20742.

Updated: 17th November 2021

Abstract

We study apps' decisions to upgrade to Android 6.0, which restricts their ability to seek blanket permissions to sensitive user information at install-time, instead requiring them to request *à la carte* permissions at runtime. Such a shift in Android's permission seeking policy comes in the wake of apps' overreaching and users' proactive measures to protect their sensitive information. Mobile apps on Android had a choice of upgrading to target Android 6.0 anytime over a three-year window instead of being forced to upgrade immediately. Given the choice of upgrading to version 6.0, that provides mobile apps with the latest platform features or staying with an earlier version that provides them with better access to user information, our study seeks to examine the upgrade decisions of apps and the outcomes of such decisions. By analyzing a unique panel dataset comprising 13,691 of the most popular apps for 24 months, we find that apps that traditionally seek more *runtime* permissions than those required for the app's functionality, delay upgrading. Specifically, we find that such upgrade delays are more likely by overreaching apps that seek to serve targeted advertisements in-app. More importantly, we find that such delaying of upgrade comes at a cost to apps in terms of marketplace outcomes such as rating and popularity. We discuss the implications of our findings for app providers as well as platform operators.

Keywords: Information Privacy, Mobile Apps, Software Upgrade, Platform Governance, Policy Change

---

Contacts: Raveesh Mayya: raveesh@stern.nyu.edu, Siva Viswanathan: sviswan1@umd.edu. The authors thank the participants of Big 10+ MIS and Analytics Conference (2019), WEIS (2019), SCECR (2019), CIST (2019), WISE (2019) and seminar participants at HKUST, CMU, LSE, UT Austin, SMU, NYU, ASU and UPitt for their comments and suggestions. Raveesh Mayya also gratefully acknowledges the financial support of the Ed Snider Center for Enterprise and Markets.

1. Introduction

The past decade has witnessed a marked rise in consumer sensitivity to online information collection and privacy. About 86% of internet users have taken steps to avoid surveillance by organizations during their online browsing sessions (Rainie et al. 2013). Recent cases of massive breach in privacy such as the Cambridge Analytica scandal (Cadwalladr and Graham-Harrison 2018) have only amplified such user concerns. With the rapid proliferation of mobile devices, privacy concerns have naturally extended to mobile devices (Barkhuus and Dey 2003). IT security researchers have shown that over a third of the smartphone apps *overreach* permissions (i.e., they gather more sensitive information from users than is strictly required for apps' functionality), which increases the risk of data misuse (Felt et al. 2011). Users have increasingly become sensitive to such practices and have been proactively taking measures to protect their privacy, including abandoning apps or platforms altogether (Pingitore et al. 2017).

As a response to these concerns and user actions, mobile platforms such as Android have changed their policies by providing users with fine-grained control over their information during their platform version upgrades. In this study, we examine one such upgrade – the release of Android's version 6.0 in late 2015. In earlier versions of Android, users had to agree to provide access to all sensitive information listed upfront by the app (such as contacts, phone memory, GPS) before downloading the app. Android 6.0 allowed users to download apps without granting any permissions and then required the apps to seek individual permissions to access users' sensitive information (viz. *runtime* permissions[1] [2]) when users used the app. Such a change in Android's privacy policy (illustrated in Figure 1) allowed users to granularly determine what permissions to grant "as they desire" [3].

[1] As per Android's website, "Runtime Permissions, also known as Dangerous permissions, give your app additional access to restricted data, and they allow your app to perform restricted actions that more substantially affect the system and other apps"
https://developer.android.com/guide/topics/permissions/overview.html#normal-dangerous
A complete list of Runtime permissions is provided in Panel A, Table A1 in the Appendix.

[2] Unlike *runtime* permissions, apps do not have to seek *install-time* permissions (e.g., phone vibration, prevent screen lock, turn on camera flash etc.). This study is about explicitly seeking *runtime* permissions.

[3] Users are not required to grant all permissions just to download the app. Instead, users can grant partial or no permissions but still download and run the app, to the extent that such permissions are not needed for its functioning.

While such changes to security and privacy policies are not new among platforms, Android's decision to provide a time-window for apps to adhere to the new policy, makes this policy change unique. Prior to Android's change, Apple introduced a fine-grained permission structure for sensitive information for iOS in 2012. Microsoft introduced User Access Control in Windows Vista OS in 2007, where programs required explicit permissions from users to access OS kernel. In both these instances, platforms used their market power to force all apps to upgrade and adhere to such policy changes if apps had to work on platforms' latest version. In contrast, Android gave mobile apps a window of three years (anytime between 2015 and 2018) to upgrade, instead of requiring them to upgrade immediately to the latest version[4]. Therefore, unlike iOS or Windows, Android allowed non-upgraded apps (i.e., apps targeting a lower Android version) to run on the latest Android versions/phones.

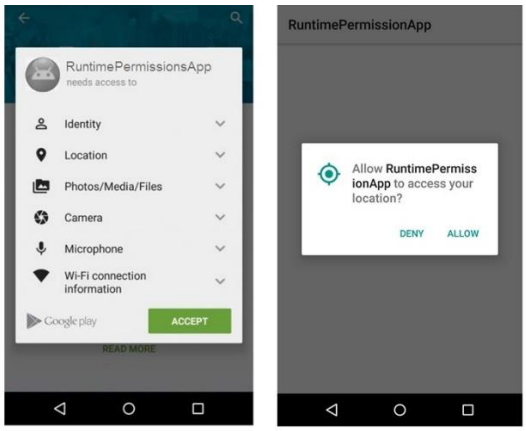


Figure 1: Change in permission seeking from download to runtime (Photo courtesy: GeneXus Community)

Android apps' choice of when to upgrade to 6.0 (anytime between 2015 and 2018) meant that apps that delayed upgrading, however, could continue to seek permissions at *install-time* (instead of at *runtime*) even when such apps were installed on latest Android versions/phones. Such apps could also update features or fix bugs using an older Android software development kits (SDKs). In other words, apps that delayed

[4] At the outset, we differentiate upgrades by users and apps. Users' upgrade means that they install the latest version of Android on their smartphones. App's upgrade means that apps compile using the latest version of the Software Development Kits (SDKs) provided by the platform and choose the latest version of the platform they want to *target* (called *targetAPI* in Android).

upgrading would be "fully functional." However, such non-upgraded apps would forgo access to the latest platform features, optimizations, and support.

Our study pursues a two-pronged empirical strategy to examine the upgrade choices made by apps and their consequent outcomes. First, study the driving factors that prompt apps to delay the upgrade using survival models. Here, we pinpoint one of the key motivations for delaying the upgrade by using text mining technique to support survival models. Second, we employ Difference-in-Difference models to quantify the effects of delaying the upgrade on apps' marketplace outcomes, i.e., apps' popularity (that dictates their ranking on the marketplace) and apps' rating. Specifically, the two research questions that this study addresses are (*i) What are the characteristics of apps that delay upgrading to seeking runtime permissions?* and *(ii) How does delaying upgrading by apps affect their marketplace outcomes (app rating and popularity)?*

We assemble a unique panel of apps between April 2016 and March 2018 by installing 16,705 most popular free Android apps on emulators and updating these apps on a monthly basis. Our main dataset consists of 13,691 of those apps that were in the Play Store for 14 months or more during the 24 months period of our dataset. Next, we develop an android app that scans the android app database of the host emulator and determines the Android version that each of the installed apps targeted, along with a list of permissions sought by the apps. Next, we collect monthly snapshot of app characteristics, such as download buckets, ratings, total comments, categories, file-size (in MB), screenshots, app description, revenue model and developers' information on over two million Android apps. We use apps' "similar apps" list on the Play Store to derive a measure of apps' popularity using weighted PageRank algorithm. Finally, we procure a unique dataset from a mobile app intelligence firm that provides us additional details for these apps from the iOS App Store (whenever the Android apps have a matching iOS app) as well as IAB categorization for the apps.

To detect apps' *overreaching*, we employ a popular statistical technique known as Peer Group Analysis (Pelikan et al. 2017; Peddinti et al. 2019) or Permission Prevalence Analysis (Taylor and Martinovic 2016). The technique uses Skip-gram Word2Vec and k-means clustering to divide apps within a category (decided

by Android) into sub-categories based on their functional similarity. The technique then, statistically determines whether each permission sought by apps is unnecessary/unusual. Employing these techniques, we distinguish permissions that are required for apps' operations/functionality (*essential* permissions) and those that are not (*nonessential* permissions). As an alternative measure, we use apps' IAB categorization to estimate *essential*/*nonessential* permissions.

As a first empirical strategy, we employ Cox Proportional Hazard models to quantify the propensity of delay caused by apps' overreaching. We find that a 1% or 1 permission increase in app's *overreaching* is associated with an increase in the propensity to delay the upgrade by 0.8% or 5.1% per month, respectively. The Kaplan-Meier estimates suggest that, on average, apps that overreach permissions delay upgrading by about three months. Next, augmenting these survival models with text mining techniques, we find that the desire to serve *in-app advertising* is a key motivation for *overreaching* which, in turn, increases their propensity to delay the upgrade. The first empirical strategy is robust to various model specifications such as (a) various operationalization of *overreaching* (e.g., absolute value of permission counts, binary variable indicating overreaching etc.) (b) mixed-effect survival analysis (that combines fixed and random effects), (c) alternative measurement of *essential/nonessential* permissions, and (d) alternative explanations such as demand side reasons or supply side cost-reasons.

Our second empirical strategy involves employing linear models with Propensity Score Matching (PSM) along with Difference-in-Differences (DiD) technique to identify the impact of delaying the upgrade. We find that apps that delay upgrading pay a penalty in terms of ratings, an equivalent of losing 1267 stars. Furthermore, delaying upgrading significantly reduces apps' popularity by 25.2%. Overall, our analyses show that delays in upgrading results in both search ranking (lower popularity) and reputational (lower rating) repercussions, both of which affect apps' overall outcomes on the platform. Our findings are robust to various specifications including a different matching procedure.

Our study contributes to the literature in multiple ways. We add to a burgeoning literature that studies the consequences of platform policy changes. In the context of digital platforms, this study is one of the earliest to analyze firm's responses when restrictive privacy policies are imposed on them. Given the

ubiquity of smartphone platforms and a number of new privacy regulations such as Europe's GDPR or California's CCPA, quantifying the antecedents and outcomes of delaying policy adoption is an important contribution to the literature. Next, the study contributes to the literature on informed consent by quantifying the unintended consequence of enabling fined-grained consent policies on sellers. Actively requesting permissions from users, while it is designed to mitigate users' privacy concerns, adds to developers' upgrade decision burden. The study also adds to the literature on platform strategy by suggesting that a fragmented platform such as Android should carefully design the upgrade window to minimize sub-optimal outcomes for the participants (apps, users) or the platform. In the same vein, the study adds to the literature on app/developers' upgrade decisions by suggesting that there is a short-term vs long-term revenue/reputational tradeoff while making upgrade decisions. Finally, our findings suggest that altering *when* users have to make information sharing decisions can help address the documented privacy paradox. By showing that even popular apps lower *overreaching* permissions, the study highlights the role of platform mechanisms in safeguarding consumer interest.

2. Theoretical Background

To place our contributions in perspective, we review the relevant literature on privacy regulation enactment, apps' overreaching behavior and upgrade decisions, identify the gaps and define our research questions.

2.1 Privacy Regulation Enactment

An active stream of literature, developed over the past decade, has investigated the impact of privacy regulations on firms' strategies and consumer welfare. The consequences of enforcing privacy policies on a firm's decision to adopt and benefit from technologies that utilize consumer's data is not straightforward. Some of the early work that focused on diffusion/adoption of information exchange systems has shown that privacy regulations have a negative effect on technology diffusion/adoption (e.g., Miller and Tucker 2009) and technology effectiveness (e.g., Goldfarb and Tucker 2011). Data regulation laws such as the GDPR can increase not just the burden on innovation and raising capital for startups (e.g., Lambrecht 2017, Jia et al. 2021) but also the market concentration of large players (e.g., Johnson et al. 2021; Peukert et al. 2020). On

the other hand, recent work has also demonstrated a more nuanced impact of privacy policies. Despite a reduction in the persistent tracking post GDPR, the inferences derived from analyzing consumer behavior data does not necessarily worsen (Aridor et al. 2020). Similarly, technology adoption can be high when privacy policies are combined with right incentives (e.g., Adjerid et al. 2016).

Related to the discussion above, economists investigating the impact of privacy policies on consumer welfare have noted that the effects can go either way. On the one hand, privacy compliance burdens could be welfare-reducing as it could drive small/specialist firms out of the market and make the market anti-competitive (Campbell et al. 2015). On the other hand, disclosure burdens could help consumers since forcing firms to disclose privacy breaches could motivate consumers to fend against theft of their sensitive information (Romanosky et al. 2011). Some of the recent studies have also noted that the privacy regulations could either help or hurt, depending on the approach that regulators follow to protect privacy (Miller and Tucker 2018). In summary, burgeoning literature has noted that privacy regulation enactment has a trade-off and that regulators should understand this tradeoff before enacting.

We add to this literature in a few ways. We are among the first to quantify the impact of privacy regulations on a multi-sided digital platform. We show that the policy designed to improve privacy choices of one side (app users) has an unintended consequence on the other side (apps) of the platform. We show that platform policy changes present a tradeoff not just to the intended side (cost of granting individual permissions versus benefits of fine-grained control), but also to the unintended side (short-term revenue loss versus medium/long-term reputational/popularity losses).

2.2 User Data, Overreaching and Platform Policy

Literature is unanimous about the importance of users' data for providers/firms. Collecting and analyzing customer data has been shown to improve product innovation (e.g., Tambe et al. 2012, Bertschek and Kesler 2017, Lambrecht 2017), improve software feature quality (e.g., Johnson et al. 2005), minimize software failures (e.g., Kechagia et al. 2015) or minimize healthcare expenses (Goldfarb and Tucker 2012). Similarly, gathering information on users' browsing behavior has been shown to significantly improve the conversion rate of internet advertisements (Goldfarb and Tucker 2011) purchase prediction (e.g.,

Montgomery et al 2004) or upsell high-margin products (e.g., Fleder and Hosanagar 2009). Consequently, there has been a marked increase in strategies that sellers/providers try to gather consumer information (e.g., Gkatzeles 2015; Savage and Waldman 2015; Kummer and Schulte 2019). Some strategies documented by smartphone app developers include lowering product price (e.g., making apps free) (Kummer and Schulte 2019), learning to seek intrusive permissions from prior app launches (Kesler et al 2017), increasing intrusive permissions across app upgrades (Wei et al 2012), focusing on specific age groups (Kesler et al 2017) or, improving market share (Kesler et al. 2019).

Related to the discussion on apps' strategy to collect users' information is the overreaching behavior, where apps collect more information than is required to provide apps' core functionality. Enabled by Android's permission seeking mechanism that was ineffective in preventing overreaching (Peng et al. 2012), over a third of Android apps have been shown to overreach (Felt et al. 2011). Given that such gathered data may be used in ways that users may not agree with (Cimpanu 2018), many security researchers have focused on designing tools that can detect and indicate overreaching (e.g., Sarma et al. 2012; Peng et al. 2012; Liu et al. 2016) including notifying users on frequency of data accessed by certain apps (e.g., Almuhimedi et al. 2015) or recommending permission settings (e.g., Liu et al. 2016).

A gap in the literature that this paper intends to close is the effectiveness of platform level policies to curb this overreaching behavior. Constant nudging can be costly and annoying (Jedrzejczyk et al. 2010) and a corresponding improvement in users' privacy behavior is uncertain (Brandimarte et al. 2013). But the negative impact on sellers is well-documented (e.g., Johnson et al. 2020). A key reason for such uncertainty in outcomes of privacy enhancing policy is the "privacy paradox", a condition where users act inconsistent to their stated desire of protecting privacy (Acquisti and Grossklags 2003, 2005, Hann et al. 2007, Spiekermann et al. 2001). Some of the documented displays of paradox are not reading the privacy policies, not going through privacy settings, or not understanding the implications of Android permissions (Hoofnagle et al. 2010, Jensen and Potts 2004, Kelley et al. 2012). In summary, platforms' actions to curb apps' intrusive data gathering strategies may result in an overall drop in innovation on the platform (Goldfarb and Tucker 2011) without a corresponding improvement in consumers' privacy behavior.

We add to this literature by first documenting a developer strategy to continue gathering users' sensitive information viz., delaying the adoption of a more privacy friendly version. Next, we add to a small but growing sub-field that studies the impact of privacy regulations on apps' demand. Finally, we add to burgeoning research on smartphone apps that has studied app demand (e.g., Ghose and Han 2014), choice of platforms (e.g., Bresnahan et al. 2014), choice of update frequency (e.g., Yin et al. 2014), entry and quality decisions (e.g., Ershov 2016), choice of promotion (e.g., Askalidis 2015; Li et al. 2016) but has not yet studied the impact of upgrade decisions on app demand under privacy policy change scenario.

2.3 Developers' Upgrade Strategies

Most studies on product upgrade strategies discuss the choice of timing by developers when releasing upgrades (e.g., Bayus 1997; Mahajan and Muller 1996). Some of the well-studied strategies include the optimal time to introduce a new (paid) version (e.g., Mahajan and Muller 1996; Wilson and Norton 1989) or the optimal discounted price that product developers can charge the consumers of an older version (Fudenberg and Tirole 1998; Fishman and Rob 2000). Even in the context of software product versioning, researchers have investigated the optimal time to upgrade and the link between upgrade intervals and market's growth rate, developer's ability to exercise planned obsolescence or to price discriminate (Fishman and Rob 2000; Mehra et al. 2013). However, in all these cases, the product upgrades are not free, and the timing strategies come with the goal of extracting surplus from consumers.

Specific to the context of free apps, recent studies have documented a correlations between upgrade frequency and demand (Comino et al. 2016), data gathering practices (Wei et al 2012, Xing et al. 2014, Taylor and Martinovic 2017), quality/performance (Jabr et al. 2020; Leyden 2021) or, an ability to monetize (Leyden 2019). Except for Jabr et al. (2020) and Leyden (2021) who study update decisions driven by platform's reputation mechanism, no other papers have studied upgrade decisions as a response to policies enacted by platforms/regulators. Similarly, while new papers have investigated the timing of policy adoption under external regulation setting such as GDPR (Garber 2018, Gradwohl 2018), we are not aware of similar studies on a digital platform setting, especially when the regulation comes from a platform owner.

Our work fills the gap in the app upgrade literature by documenting the consequences of delaying upgrading to continue harvesting users' sensitive information.

2.4 Research Questions

We find several gaps in the literature that informs our understanding of the antecedents and consequences of delaying privacy policy adoption on platforms. First, there is a dearth of research examining factors that drive some apps' upgrade decisions under a new privacy policy. Hence, our first research question (RQ1) asks and answers, *"What are the characteristics of apps that delay upgrading to seeking runtime permissions?"*. We premise this question using the insights from literature on apps' *overreaching*. The shift from blanket permissions at download-time to *à la carte* permissions at *runtime* focuses users' attention on individual permissions sought by apps. Such enhanced attention on permissions may drive *overreaching* apps to delay upgrading to the latest version. Further, we dive into identifying a key reason for why apps delay upgrading. Depending on how much the app relies on harvesting users' sensitive data for monetization, upgrading to the latest version has different associated costs and benefits for apps with different revenue models. Based on this cost-benefit calculus logic, we expect apps whose revenue models benefit from *overreaching* user permissions to delay the upgrade. Specifically, we investigate how *in-app-advertising* as a revenue model affects apps' overreaching decisions which, in turn, affects apps' decision to delay upgrade.

Our second research question (RQ2) seeks to empirically identify the outcomes associated with delaying the upgrade. Specifically, we ask and answer, "*How does apps' decision to delay upgrading affect their marketplace outcomes (rating and popularity)?".* Here, the focus is on quantifying the negative outcomes from delaying upgrading to Android 6.0. While the benefits of delaying the upgrade is clear (continued access to users' sensitive information), existing research on privacy policy enactment and on upgrade strategies does not highlight if there are any disadvantages in delaying the upgrade. Quantifying the repercussions, if any, of deliberately delaying upgrading has important policy implications for an open platform such as Android. The next section describes our research context and data to address the above-mentioned questions.

3. Research Context and Data

We examine our research questions on the Android Play Store, the app marketplace of the largest smartphone platform with over 80% market share (Statista 2021a). Android releases platform-level upgrades each year (also called version upgrades) for both end users and developers. Our research focuses on the launch of version 6.0 (codenamed Marshmallow) in late 2015 that was akin to the earlier version launches but with a major restriction on data gathering. Similar to earlier upgrades, Android 6.0 brought feature and aesthetic improvements such as notification improvements (see Panel B and C of Appendix Table A1). But unlike earlier upgrades that only introduced feature additions and improvements, Android 6.0 also introduced a new restriction that altered how apps sought permissions [5]. Coupled with a liberal timeline (of 3 years) to adopt this restriction, this provides us a natural experiment to study the impact of privacy regulations on seller (i.e., apps) behavior on a digital platform. Our panel starts from April 2016 when the user installation base for Android 6.0 reached 5% (Android 2016, Statista 2021b) [6] and ends 6 months before apps are required to mandatorily adopt the privacy restriction.

We compiled our data by combining three sources that we gathered over two years. First, we created a list of 21,000 [7] most downloaded free/freemium apps[8] as of April 2016 and installed these apps (more than one terabyte) on emulators on research PCs at our lab[9]. We ensured that the dataset is quite large and the distribution is representative of the global distribution (see online appendix C for comparison). Installing apps is the only clean way to know which Android version that app targeted since only an installed file carries the signature (SDK version) with the required information (called *targetAPI*). Once the apps were

5 A careful study of Appendix Table A1 suggests that sensitive information access restrictions are the only new restrictions that the platform introduced.

6 Examining the data from April 2016 addresses the potential lack of motivation among apps in the initial months when there were not many Android users who had upgraded to Android 6.0. Initial 5% of users have been shown to exhibit a willingness to take risks with new features and provide valuable feedback to products for further improvement (Mahajan and Muller 1998), thereby helping app's evolution.

7 The list contains all apps that had greater than 500,000 downloads (Around 18,600 apps) and about 2400 apps with 100,000-500,000 or lower downloads.

8 We choose free apps instead of paid apps because paid apps represent only 3.3% of the apps in the Play Store (Statista link). Paid apps also tend to be less strategic in terms of permissions, due to lower reliance on user's information for revenues.

9 The emulator software emulated Samsung Galaxy Tab 3 (10.1 inch), a popular tablet device around 2016.

installed, we developed and installed a custom-built Android app that scanned all installed apps and extracted the *targetAPI* of each installed app, the app version number, and a list of all sought permissions. We updated all installed apps monthly and collected the above-mentioned data points for 24 months. Our second data source is Android's Play Store website from which we exhaustively collected details of over two million apps on a monthly basis. We collected information such as app description, rating, download count bucket, count of reviews, categories, date of last update, number of screenshots uploaded, app's file size (in MB), developer ID and two dummy variables, one each to indicate whether the app has *in-app advertising* or *in-app purchase* options for each month. Third, we procured a proprietary dataset from an app intelligence firm from which we got the iOS App Store details (if available) for the Android apps in our dataset as well as the IAB categorization of the apps[10]. Merging the three datasets gave us an unbalanced panel of 16, 976 [11] apps. We further dropped 3015 apps from our analysis as they were pulled out of the Play Store in less than 14 months. The resultant panel consists of 13,691[12] apps over 24 months (April 2016 to March 2018), resulting in 325,548 app-month observations [13]. Within the 13,691 apps, 2016 had upgraded to Android 6.0 before April 2016. While we retain them for estimating the *essential/nonessential* permissions list, we only use 11,645 apps in our analyses (a total of 277,743 observations). Appendix Table C1 summarizes how we filtered out the apps.

3.1 Detecting Overreaching of Apps

Central to our investigation is detecting *overreaching*, i.e., which apps seek permissions that are not necessary to their core functionality which we call *nonessential* [14] permissions (Jana et al. 2015). Our main

10 This additional dataset is used for robustness checks and testing the sensitivity of our main results.

[11] Of the list of 21000 apps, 4,295 apps were not available to download on emulators in the United States. 709 apps that were pulled out of the Play Store had more than or equal to 14 months of data, so we have retained them in the analysis.

[12] Of the list of 21000 apps, 4,295 apps were not available to download on emulators in the United States. 709 apps that were pulled out of the Play Store had more than or equal to 14 months of data, so we have retained them in the analysis.

[13] Some apps are not available in Play Store for all 24 months, hence the unbalanced panel does not have 13,691*24 observations.

[14] Academics and practitioners (e.g., Jana et al. 2015, Taylor and Martinovic 2016; Pelikan et al. 2017; Peddinti et al. 2019) agree about categorizing permissions that are not required to provide such services, as *nonessential*. If a permission is required for app's monetization, survival, analytics or personalization but not to provide core functionality, it is to be categorized as *nonessential*/unnecessary.

analysis follows well-known statistical techniques popularized by security researchers: Peer Group Analysis (Pelikan et al. 2017; Peddinti et al. 2019) or Permission Prevalence Analysis (Taylor and Martinovic 2016). These statistical techniques require apps to be sub-grouped into functionally similar apps. Once the apps are grouped, a majority/cutoff polling is used to determine (within a functionally similar group) if the app is seeking unusual permissions. Figure 2 graphically represents the two-step procedure to detect overreaching.

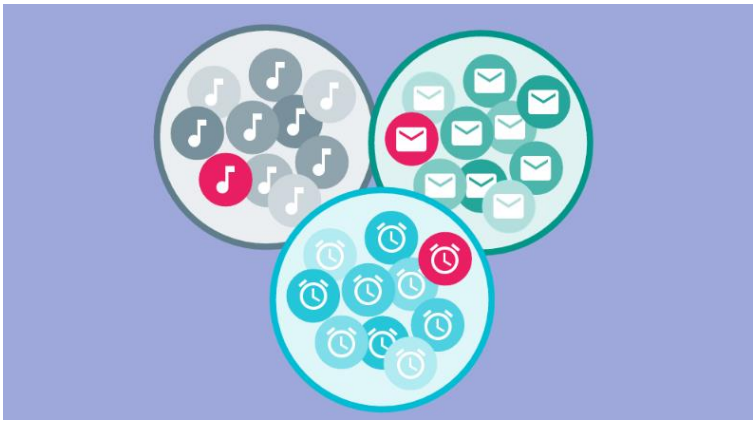


Figure 2: Peer Group Analysis (Image courtesy: Pelikan et al. 2017)

Our main analysis uses a common technique for such grouping: employing vector-embedding techniques (Pelikan et al. 2017; Peddinti et al. 2019). We employed Skip-Gram Word2Vec, a popular vector embedding technique, to categorize apps based on their text description. A distributed vector representation of words (e.g., Word2Vec) learns the locational similarity and context of words in a statement (synonyms, contexts of words or grammatical nuances such as plurals) and has been proven to be more accurate than the traditional bag of words or n-gram vectors (Mnih and Hinton 2009). Furthermore, traditional techniques such as TF-IDF are known to suffer from *curse of dimensionality* when the corpus is large (Hinneburg and Keim 1999). Word2Vec by Mikolov et al. (2013) resolves these issues by using the Skip-Gram model to create a 300-dimensional distributed vector representation for each word. We obtain a single 300-dimensional vector for a document by combining individual word vectors of that document. Since app descriptions are written in standard English with carefully curated English words/phrases, we use a model

trained on 100 billion words from a Google News dataset to improve the accuracy of dividing apps into functionally similar peer groups.

We start by tagging each app description from the Play Store with a category label determined by Android (see Panel A of Table A2 for a comprehensive list of categories). Since Android pre-determines these categories, we are assured of the general similarity between apps under the same category. Next, we use an unsupervised learning technique to divide apps within each category into sub-categories with similar functionality and utility.

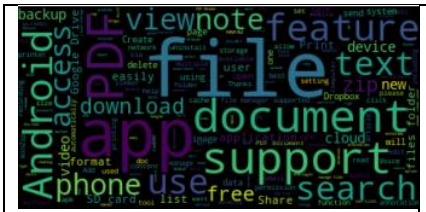
Codifying the process to determine the sub-category of apps, we:

I. Obtain a 300-dimensional vector for each app based on app description by employing Word2Vec.

II. Employ k-means clustering to determine the optimal sub-categories within each category as follows:

a. Determine the range of optimal clusters [k_m-k_n] for app category *c* using Elbow technique, AIC, BIC
b. Determine the least number of clusters (k_c), within [k_m-k_n], wherein each cluster has at least five apps.
c. Divide apps under each category c into k_c clusters (i.e., k_c sub-categories of each of *c* category)

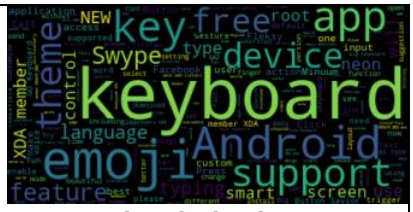
This process provides us a total of 411 sub-categories. To illustrate the effectiveness of sub-categorization, we chart a word-cloud for the sub-categories created under two of the most generic categories, "Productivity" and "Lifestyle." As shown in Figures 3A and 3B, the sub-categories are both homogenous and distinct from each other. We also manually verify a random sub-set of all algorithmically generated sub-categories to ensure that the sub-categorization has worked well.



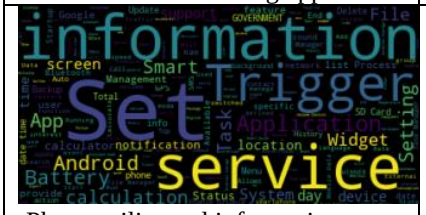
Document editing apps



Calendar-related apps



Keyboard related apps

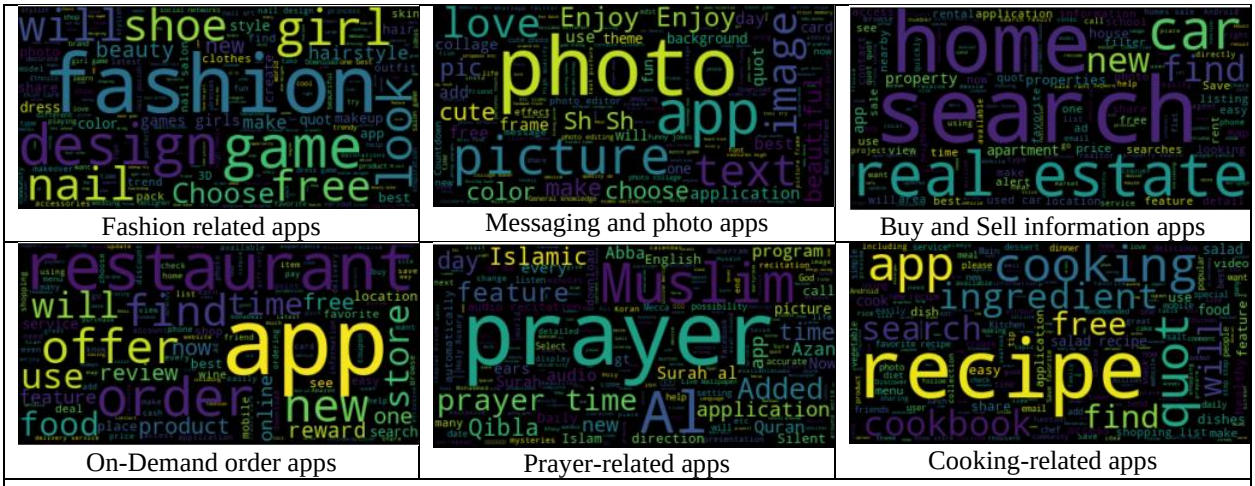


Phone utility and information apps



Productivity enhancing apps

Figure 3A: Distribution of words in sub-categories created from category "Productivity"



Fashion related apps | Messaging and photo apps | Buy and Sell information apps

On-Demand order apps | Prayer-related apps | Cooking-related apps

Figure 3B: Distribution of words in sub-categories created from category "Lifestyle"

Next, following cutoff-polling technique (e.g., Sarma et al. 2012; Taylor and Martinovic 2016), we determine which of the permissions sought are *essential* to all apps in each sub-category/peer group. We code those permissions requested by more than 80% of the apps in a sub-category/peer group as *essential* permissions. The rest of the permissions, idiosyncratic to specific apps in a group, are tagged as *nonessential* (Erlingsson 2016). This process provides a list of *essential* and *nonessential* permissions for each of the 13,691 apps and such list is populated independently for each of the 24 months. To build intuition, consider navigation apps: permission to access GPS is essential for any navigation app, hence at least 8/10 of such apps would seek permissions to GPS. In contrast, access to users' call log does not seem essential to navigation apps; hence we would expect less than 8/10 of the apps to seek call log permission. We test the sensitivity of cutoff by varying the threshold between 70% and 90%.

The unsupervised technique described above is prone to misclassification of categorization. Despite manual verification and utmost effort to ensure proper classification, it is still likely that some apps are grouped with unrelated apps due to specific words/phrases used in the text description. Hence, we use apps' Interactive Advertising Board (IAB) categorization to obtain the alternative measure of *essential/nonessential* classification. While submitting the apps to the Play Store, apps declare their IAB category, following a well-defined and highly granular IAB taxonomy (IAB 2021). Accurately declaring this taxonomy is essential for contextual targeting on the internet, hence these apps have high motivation

to declare accurately. We obtain apps' IAB categories for all apps in our dataset from a mobile app intelligence firm.[15] This process classifies apps into one of 120 IAB categories. Once we obtain IAB categories, we perform the cutoff polling technique (as above) on each IAB category to get *essential/nonessential* classification.

3.2 Popularity of Apps

The Play Store does not publicly disclose the exact download count for each app. This non-disclosure precludes researchers from using exact download counts to estimate an app's popularity. Hence, we determine app popularity by building a network of apps from the "similar apps" list and estimating each app's network centrality. Specifically, we employ the Weighted PageRank algorithm (Page et al. 1999) to derive apps' popularity score. Researchers (e.g., Oestreicher-Singer and Sundararajan 2012; Kane and Ransbotham 2016) have employed variants of PageRank algorithm to derive the importance of various nodes in a network using the concept of the *eigenvector centrality*.

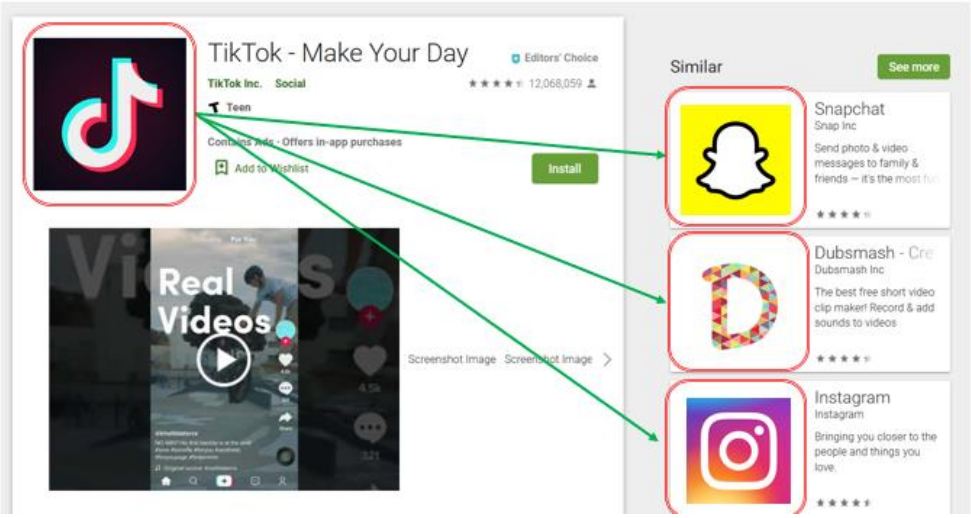


Figure 4: Screenshot of the Android Play Store where each app displays the list of "similar apps".

Note: Dubsmash appears above Instagram since it is (or, used to be) a more popular video creation/sharing app than Instagram. That said, Instagram appears in "similar apps" list of not just video but also image and general social media apps, hence the *eigenvector centrality* score of Instagram is higher than Dubsmash.

Intuitively, an influential node has a larger *eigenvector centrality*. Alternatively, a larger value of *eigenvector centrality* for a node also means that the node is pointed to by (i.e., receives a vote of support

15 The firm also employs NLP techniques to predict app's IAB category and compares it with the declared category. In rare instances where the app has declared the IAB category wrong, the firm provides the predicted category.

from) many nodes or more popular nodes. We employ the latter interpretation of *eigenvector centrality* in our investigation.

For each app in the Play Store, Android provides a list of "similar" apps (see Figure 4). A similar-app list can be thought of as a "vote of support" from the focal app to similar apps (Oestreicher-Singer and Sundararajan 2012). Intuitively, the greater the popularity of an app, the greater the likelihood that such an app would appear on similar apps' list of many other apps in related categories. Therefore, by gathering similar apps' lists for each app, we can reverse-engineer how popular any app is. Specifically, we use a modified variant of PageRank algorithm, which uses an app's position in the "similar" list to determine the strength of the vote of support [16]. We repeat this process for each month to estimate an app's popularity in any given month.

3.3 Variable Definitions

Table 1 provides a detailed description of the variables used in the major analyses. Table 2 provides the summary statistics and correlation matrix of the variables.

Variables of Interest: The first four variables of interest are the count of total *runtime* permissions, *install-time* permissions, *essential* permissions, and *nonessential* permissions sought. We use these four count variables to construct the three normalized variables: ratio of *runtime*, *essential* and *nonessential* permissions sought to all permissions used by an app (*runtime*+*install-time*) and use the three ratios as independent variables in survival models[17]. The dependent variable in survival models, $upgrade_{it}$ is coded as one after app *i* performs an upgrade to the latest version and 0 before that. The upgrade from version 5.1 (or below) to 6.0 (or above) is irreversible, so once upgraded, the value stays at 1. Variable $rating_{it}$ captures the cumulative user rating that the app *i* has received by the end of month *t*. Finally, $popularity_{it}$ is obtained using weighted PageRank algorithm as outlined in Section 3.2.

[16] we use the formula *max(1-(rank/24), 0.25)* to determine the strength of the directed edge from the focal app to similar apps

[17] We re-estimate all models with count variables (instead of ratios) and find qualitatively consistent results.

Table 1. Variable Explanation

Variable	Description
Key Variables	
$runtime_permissions_count_{it}$	Count of *runtime* permissions sought by app *i* in month *t*
$installtime_permissions_count_{it}$ [18]	Count of *install-time* permissions declared by app *i* in month *t*
$essential_\ permissions_count_{it}$	Count of *essential* runtime permissions sought by app *i* in month *t*
$nonessential_\ permissions_count_{it}$	Count of *nonessential* runtime permissions sought by app *i* in month *t*
Dependent Variables	
$runtime_permissions_{it}$	Ratio of runtime to all permissions sought by app *i* in month *t*, defined as ($runtime_perm_count_{it}$) / ($runtime_perm_count_{it}$ + $installtime_perm_count_{it}$)
$essential_\ permissions_{it}$	Ratio of *essential* permissions to all permissions sought by app *i* in month *t*
$nonessential_\ permissions_{it}$	Ratio of *nonessential* to *all* permissions sought by app *i* in month *t*
$upgrade_{it}$	A dummy variable indicating if the app *i* upgrades to Android 6.0 (or above) in the month *t*
$rating_{it}$	The cumulative user rating of app *i* by the end of month *t*
$popularity_{it}$	The popularity of app *i* by the end of month *t*, computed using the weighted PageRank algorithm
App Characteristics	
$rating_count_{it}$	Total count of ratings given by raters for app *i* by the end of month *t*
$dayssinceupdate_{it}$	Count of days since updating app *i* as measured at the end of month *t*
$screenshots_{it}$	Count of screenshots [19] (including optional video) uploaded by app *i* to Play Store by the end of month *t*
$developer_appcount_{it}$	Number of apps in Play Store by the developer of app i of month *t*
$filesize_{it}$	File size (in MB) of app *i* by the end of month *t*
$inAppAdvertising_{it}$	A dummy variable indicating whether app *i* displays ads in the month *t*
$inAppPurchases_{it}$	A dummy variable indicating whether app *i* allows in-app purchases in the month *t*
Control Variables	
$category_group_i$	The app category group that app *i* belongs to
$download_bucket_{it}$ [20]	Download bucket (group) that app *i* belongs to at the end of month *t*
$month_dummy_t$	A vector of dummies that represents if month *t* is April-2016, May-2017, etc

Note: Missing values in file sizes (such as "varies with devices" or months when file sizes were not displayed in the Play Store) are handled by replacing them with values carried forward from previous months. If values are still missing, we replace it with mean value of file-size of that app. If still missing, they are replaced by average file-sizes of all apps in a sub-category in month t.

Explanatory Variables: App characteristics that represent app demand, appearance on the Play Store and developers' details are used as explanatory variables. For each app in a given month, we capture the total count of ratings provided by users, the count of days elapsed since the developer has pushed an update

[18] Install-time permissions cover areas which pose very little risk to user's privacy (e.g., setting alarm, accessing Wi-Fi state). Consumers don't have a choice to rejecting them since developers are not required to explicitly seek permissions from users. https://developer.android.com/guide/topics/permissions/overview.html#normal-dangerous

[19] Play Store mandates that a minimum of 2 and a maximum of 8 screenshots per target category (phone, tablet, TV and Wear OS) may be uploaded on the Play Store. If the developer has uploaded a video, we count it as a screenshot. https://support.google.com/googleplay/android-developer/answer/1078870?hl=en

[20] The download buckets: 1-5 downloads, 5-10, 10-50, 50-100, 100-500, 500-1k, 1k-5k, 5k-10k, 10k-50k, 50k-100k, 100k-500k, 500k-1M, 1M-5M, 5M-10M, 10M-50M, 50M-100M, 100M-500M, 500M-1B and 1B-5B downloads

(features, bug fixes or upgrades), total count of apps by focal app's developer on the entire Play Store, count of screenshots on an app's Play Store page, file-size and revenue model (*in-app-purchases* or *in-app-advertising*).

Table 2(a). Summary Statistics

	13,691 apps in the dataset					11,675 apps in main analyses		
Variables	**# Obs.**	**Mean**	**Std. Dev.**	**Min**	**Max**	**# Obs.**	**Mean**	**Std. Dev.**
Key Variables								
$runtime_permissions_count_{it}$	325,548	3.49	2.67	0	22	277,743	3.38	2.56
$installtime_permissions_count_{it}$	325,548	8.22	4.70	0	65	277,743	7.99	4.36
$essential_ permissions_count_{it}$	325,548	0.54	0.69	0	10	277,743	0.51	0.67
$nonessential_ permissions_count_{it}$	325,548	2.95	2.46	0	20	277,743	2.86	2.37
$runtime_permissions_{it}$	325,548	26.49	13.22	0	100	277,743	26.21	13.38
$essential_permissions_{it}$	325,548	4.42	5.79	0	100	277,743	4.23	5.72
$nonessential_permissions_{it}$	325,548	22.07	12.55	0	100	277,743	21.97	12.73
$upgrade_{it}$	325,548	0.54	0.50	0	1	277,743	0.46	0.50
$rating_{it}$	325,548	4.04	0.37	1.86	4.96	277,743	4.03	0.38
$popularity_{it}$	325,548	8.88	15.10	0.00	593.24	277,743	7.91	12.98
App Characteristics								
$rating_count_{it}$	325,548	140772.4	1050150	35	75718104	277,743	86019.52	918514.20
$dayssinceupdate_{it}$	325,548	151.37	175.57	0	2799	277,743	153.39	175.20
$screenshots_{it}$	325,548	12.46	5.47	2	33	277,743	12.39	5.35
$filesize_{it}$	325,548	24.21	26.51	0.024	971	277,743	24.01	25.73
$developer_appcount_{it}$	325,548	28.33	84.46	1	1914	277,743	28.54	87.51
$inAppAdvertising_{it}$	325,548	0.13	0.34	0	1	277,743	0.13	0.33
$inAppPurchases_{it}$	325,548	0.28	0.45	0	1	277,743	0.26	0.44

Note: These statistics are based on the sample before propensity score matching.
Since the computed PageRank scores are numerically small, $popularity_{it}$ is scaled by a factor of 1 million.

Table 2(b). Correlation Matrix

	Variables	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**	**(7)**	**(8)**	**(9)**	**(10)**	**(11)**	**(12)**	**(1)**	**(14)**	**(15)**	**(16)**	**(17)**
(1)	$runtime_permissions_count_{it}$	1																
(2)	$installtime_permissions_count_{it}$	**0.76**	1															
(3)	$essential_ permissions_count_{it}$	**0.42**	**0.33**	1														
(4)	$nonessential_ permissions_count_{it}$	**0.97**	**0.73**	**0.17**	1													
(5)	$runtime_permissions_{it}$	**0.67**	**0.24**	**0.34**	**0.63**	1												
(6)	$essential_ permissions_{it}$	**0.14**	**0.02**	**0.85**	**-0.09**	**0.33**	1											
(7)	$nonessential_ permissions_{it}$	**0.64**	**0.25**	**-0.03**	**0.70**	**0.90**	**-0.11**	1										
(8)	$upgrade_{it}$	**0.01**	**0.07**	**0.10**	**-0.01**	**-0.04**	**0.08**	**-0.07**	1									
(9)	$rating_{it}$	**0.09**	**0.18**	**0.08**	**0.08**	**-0.01**	**0.02**	**-0.02**	**0.12**	1								
(10)	$popularity_{it}$	**0.08**	**0.13**	**0.06**	**0.07**	**0.01**	**0.03**	**0.00**	**0.16**	**0.24**	1							
(11)	$rating_count_{it}$	**0.11**	**0.17**	**0.05**	**0.11**	**0.00**	0.00	0.00	**0.08**	**0.11**	**0.27**	1						
(12)	$dayssinceupdate_{it}$	**-0.17**	**-0.19**	**-0.10**	**-0.16**	**-0.07**	**-0.02**	**-0.06**	**-0.20**	**-0.15**	**-0.16**	**-0.06**	1					
(13)	$screenshots_{it}$	**-0.12**	**-0.08**	**-0.06**	**-0.11**	**-0.05**	**-0.02**	**-0.04**	**0.05**	**0.04**	**0.07**	**0.02**	**0.01**	1				
(14)	$filesize_{it}$	**-0.01**	**0.01**	**0.02**	**-0.02**	**0.02**	**0.03**	**0.01**	**0.05**	**0.04**	**0.05**	**0.07**	**0.03**	**0.31**	1			
(15)	$developer_appcount_{it}$	**-0.09**	**-0.03**	0.00	**-0.10**	**-0.07**	**0.02**	**-0.08**	**-0.03**	**-0.03**	**-0.04**	**-0.02**	**0.07**	**0.01**	**0.02**	1		
(16)	$inAppAdvertising_{it}$	**-0.01**	**0.03**	0.00	**-0.01**	**-0.04**	0.00	**-0.04**	**0.11**	**0.06**	**0.11**	**0.06**	**0.07**	**0.03**	**0.04**	0.00	1	
(17)	$inAppPurchases_{it}$	**0.04**	**0.10**	**0.04**	**0.03**	**0.01**	**0.03**	0.00	**0.13**	**0.18**	**0.14**	**0.06**	**0.01**	**0.22**	**0.28**	**-0.03**	**0.19**	1

Control Variables: We include app category and download bucket dummies. Based on prior research (e.g., Ghose and Han 2014) and how categories evolve, apps are divided into six categories: Online Content Consumption (Media and Entertainment), Learn and Explore, Personal (Social and Lifestyle), Mobile Specific Utilities (services that are enabled by a cell phone technology such as GPS), Mobile Access Utilities (mobile apps provided by offline or internet utility firms), and Games.[21] Panel A of Appendix Table A2 provides more details on the category groups and categories. Variable $download_bucket_{it}$ captures the range of app downloads that Play Store publishes.

4. Analyses and Results

4.1 RQ1: What are the characteristics of apps that delay upgrading to seeking runtime permissions?

Our first empirical strategy is to explore the antecedents of delaying the upgrade to Android 6.0. For this analysis, we employ a continuous-time single-failure survival model. Specifically, we employ the Cox Proportional Hazard model (e.g., Adomopoulos et al 2018; Bapna et al. 2016; Kumar et al. 2014) that investigates how various covariates affect apps' decisions to delay upgrading to the latest version of Android. The model is specified as below:

$$\boldsymbol{\lambda_i(t)} = \lambda_0(t)\exp(c_1 * essential_permissions_i + c_2 * nonessential_permissions_i + c_3 * rating_i + c_4 * log(popularity_i) + c_5 * \Delta rating_count_i + c_6 * dayssinceupdate_i + c_7 * screenshots_i + c_8 * log(filesize_i) + c_9 * developer_appcount_i + \Gamma * download_bucket_i + \Lambda * category_group_i) \quad (1)$$

where the subscript i indexes apps and subscript t indexes months. $\lambda_i(t)$ is the hazard of the app i upgrading to Android 6.0 at time t and $\lambda_0(t)$ represents the baseline hazard of upgrading [22]. A hazard rate of less than 1 (HR<1) indicates that the covariate decelerates the chances of failure (in this study, upgrading to Android 6.0 is *failure*), i.e., increases the chances of delaying the upgrade. Such an interpretation is consistent with the literature (e.g., Kauffman et al. 2000). We also re-estimate models by replacing the main

[21] In an unreported tests, we replace the six categories above with the sub-categories obtained using Peer Group Analysis and IAB categories to test the sensitivity and find consistent results.

[22] Single-failure specification discards the observations after *failure* (upgrade) has occurred. Hence, out of 277,743 observations, only 156,903 observations are retained in the survival analysis. See Appendix Figure A1 for a sample demonstration.

dependent variables (ratios) with count values and include $installtime_permissions_count_{it}$ as control. Finally, to get an intuitive sense of the impact of overreaching, we re-estimate model (1) by replacing variables representing essential/nonessential permissions with a single dichotomous variable $overreached_i$, which is a dummy variable that carries the value of 1 if an app seeks at least one *nonessential* permission (indicative of an app overreaching permissions). As in the earlier case, if we observe HR<1, we can infer that overreaching apps have a higher propensity to delay upgrading to Android 6.0. To get an intuitive sense of the impact of overreaching on upgrade decisions, we chart the Kaplan-Meier curve contrasting the "survival" (i.e., delaying upgrade) of overreaching and non-overreaching apps. As shown in Figure 5, overreaching apps have a higher chance of "survival" (i.e., delaying upgrade) compared to those that do not overreach. Importantly, the graph shows that the difference in median "survival" is about three months for treated apps, meaning that apps that overreach, delay upgrading by three months, on average.

A limitation of Cox proportional hazard models is their limited ability to account for heterogeneity across apps by means of fixed effects. Despite the popularity of Cox PH models without fixed effects across disciplines: IS (e.g., Adomopoulos et al 2018; Bapna et al. 2016; Kumar et al. 2014), Marketing (e.g., Jain and Vilcassim 1991; Lambrecht and Tucker 2013; Seetharaman and Chintagunta 2003), Economics (e.g., Brown and Dinc 2005; Light 1995; Towe and Lawley 2013) among others, CoxPH is limited in its interpretation. Hence, we employ multi-level (i.e., mixed-effects) survival models as alternative model specification. Mixed effect models combine fixed and random effects specifications that allow for both time-variant and time-invariant covariates in the same model (Austin 2017). While such models are really popular in bio-medical research, these models are becoming relevant in IS literature as well (e.g., Angst et al. 2017). Since our outcome variable is repeatedly measured within an app, we specify an app-level nesting and fit the model using Weibull distribution. Such a specification allows us to estimate a within-app variation as well as to have random effects for the intercept and slope of each app.

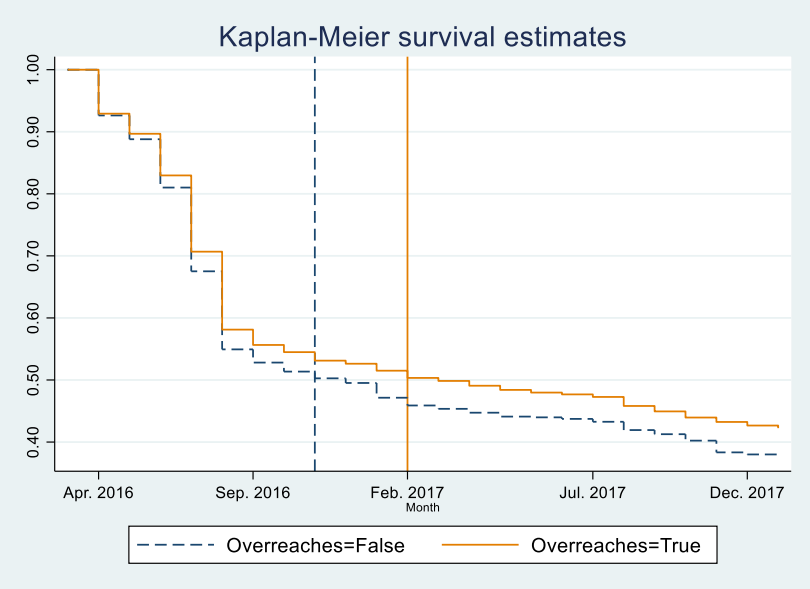


Figure 5: Kaplan-Meier graph comparing overreaching apps with those that do not. Vertical lines indicate the median survival rate. On average, apps that overreach delay the upgrade by three months (Feb 2017 vis-à-vis November 2016).

The results of Cox PH model and Mixed effect survival model estimations are presented in Table 3. From column (1) of Table 3, we find that the Hazard Ratio of nonessential permissions are less than 1, suggesting that apps' overreaching behavior predicts apps' upgrading delays. A HR of 0.992 means that a 1-unit increase (1% increase in ratio) predicts a delay in the upgrade by 0.8% ((1-0.992)*100=0.8%) per month. Similarly, Column (3) of Table 3 shows that one additional *nonessential* permission increases the propensity to delay upgrading by 5.1% per month. On the other hand, the HR of essential permissions is either greater than 1 or is insignificant, indicating that essential permissions do not influence apps' decision to delay upgrading. Our arguments are further supported by the model where we replace *runtime* permissions variable with *overreached* variable. Column (5) of Table 3 suggests that overreaching increases the propensity to delay upgrading by 24.6% ((1-0.754)*100=24.6%). Overall, the models quantitatively establish that overreaching is associated with apps' decision to delay upgrading. The mixed effect models (Columns 2,4 and 6) provide consistent and numerically larger results as Cox Proportional Models. Finally, Appendix Table A3 shows results when all runtime permissions are pooled together. We find that the results are consistent (i.e., an increase in runtime permissions increases the propensity to delay upgrade).

We shift our focus on other app characteristics to understand the heterogeneity in outcomes. First, the HR of $rating_{it}$ is insignificant. These suggest that app's quality (i.e., rating) do not influence the decision to delay the upgrade. Days since the latest update, an indicator of app's maintenance cycle, increases the propensity to delay the upgrade. On the other hand, app's file-size, an indicator of the overall functionality provided by the app (Ghose and Han 2014) does not increase apps' propensity to delay the upgrade. In summary, apps that are sporadically maintained or those that provide specific/minimal functionalities are more likely to delay the upgrade (a thorough discussion is presented in Section 5.1). Finally, compared to the "online content consumption" apps (e.g., video or audio streaming services), mobile-specific utility apps (e.g., messaging, navigation, food delivery apps), mobile access apps (e.g., thermostat, shopping mall, bank app, ticket booking) and personal apps (such as social networking/dating apps) are more likely to delay the upgrade, consistent with reports in popular press (Maheshwari 2017, Stamm et al. 2018).

In summary, the analyses suggest that apps that *overreach* are more likely to delay upgrading to a privacy regime where apps have to explicitly seeking individual permissions during runtime.

4.2 Robustness Checks

We take two distinct approaches to check the sensitivity of our finding. First, we attempt to rule-out alternative explanations using different sub-sample analyses. Second, we re-estimate the models using an alternative measurement of peer group using IAB categorization.

4.2.1 Alternative Explanations

4.2.1.1 Supply Side Operational Reasons: Developers incur certain cost to upgrade (e.g., work force training, change in app control flow) and such costs depend on either developer's size/capacity or apps' supported functionalities. Furthermore, the permission seeking regime changed from download time to run-time, hence apps could no longer assume that all permissions were available. This would necessitate a re-wiring of the logic and consequently result in an increase in the time needed to upgrade. The burden of re-wiring the logic is more for apps that seek a lot of permissions and hence this correlation.

Table 3. Analysis of Upgrading to Latest Version of Android

Dependent variable - *upgrade*$_{it}$	Model (1)		Model (2)		Model (3)		Model (4)		Model (5)		Model (6)	
Model Variant	*Cox PH Model*		*Mixed Effects Survival Model*		*Cox PH Model*		*Mixed Effects Survival Model*		*Cox PH Model*		*Mixed Effects Survival Model*	
	Hazard Ratio		Hazard Ratio		Hazard Ratio		Hazard Ratio		Hazard Ratio		Hazard Ratio	
essential_ permissions$_{it}$	1.004*	(0.002)	1.062***	(0.003)								
nonessential_ permissions$_{it}$	0.992***	(0.001)	0.982***	(0.001)								
essential_ permissions_count$_{it}$					0.992	(0.018)	1.661***	(0.044)				
nonessential_ permissions_count$_{it}$					0.949***	(0.007)	0.863***	(0.010)				
overreached$_i$ *(1/0)*									0.754***	(0.021)	0.600***	(0.024)
Δrating_count$_{it}$	1.000	(0.000)	1.000***	(0.000)	1.000	(0.000)	1.000***	(0.000)	1.000	(0.000)	1.000***	(0.000)
rating$_{it}$	0.978	(0.031)	1.014	(0.049)	1.034	(0.033)	1.051	(0.051)	1.006	(0.032)	1.022	(0.049)
log(popularity$_{it}$*)*	0.979	(0.011)	1.207***	(0.018)	0.966**	(0.011)	1.201***	(0.018)	0.977*	(0.011)	1.219***	(0.018)
games$_i$	0.939	(0.038)	1.060	(0.064)	0.840***	(0.034)	0.934	(0.057)	0.907*	(0.037)	0.954	(0.058)
personal_apps$_i$	0.888**	(0.041)	0.989	(0.067)	0.914*	(0.042)	1.026	(0.070)	0.880**	(0.040)	0.991	(0.067)
utility_mobilespecific$_i$	0.829***	(0.037)	0.744***	(0.050)	0.887**	(0.039)	0.814**	(0.055)	0.823***	(0.037)	0.764***	(0.051)
utility_mobileaccess$_i$	0.756***	(0.039)	0.792**	(0.064)	0.836***	(0.044)	0.868	(0.072)	0.755***	(0.039)	0.816*	(0.065)
learn_explore$_i$	0.962	(0.043)	1.018	(0.069)	0.949	(0.043)	1.001	(0.069)	0.951	(0.043)	0.994	(0.068)
category$_i$*= content_consumption*$_i$ *(baseline)*												
dayssinceupdate$_{it}$	0.954***	(0.001)	0.956***	(0.001)	0.954***	(0.001)	0.956***	(0.001)	0.954***	(0.001)	0.955***	(0.001)
screenshots$_{it}$	1.003	(0.002)	1.020***	(0.004)	1.001	(0.002)	1.019***	(0.004)	1.003	(0.002)	1.023***	(0.004)
log(filesize$_{it}$*)*	1.068***	(0.017)	1.141***	(0.027)	1.123***	(0.018)	1.199***	(0.028)	1.081***	(0.017)	1.194***	(0.028)
developer_appcount$_{it}$	1.000	(0.000)	0.998***	(0.000)	1.000*	(0.000)	0.998***	(0.000)	1.000	(0.000)	0.998***	(0.000)
installtime_permissions_count$_{it}$					0.971***	(0.004)	0.971***	(0.006)				
ln(p)			161.709***	(1.494)			162.399***	(1.494)			172.129***	(1.594)
var(packageid)			163.946***	(27.673)			156.979***	(26.379)			169.910***	(29.862)
Count of Apps	11,675		11,675		11,675		11,675		11,675		11,675	
Download Bucket Dummy	YES		YES		YES		YES		YES		YES	
Log Likelihood	-56285.88		-25197.2		-56170.48		-25155.82		-56279.37		-25438.84	
AIC	112607.8		50436.4		112379		50355.65		112592.7		50917.67	
BIC	112787.1		50645.63		112568.3		50574.84		112762.1		51116.94	
# of Obs.	156,903		156,903		156,903		156,903		156,903		156,903	
X^2	2245.11		2973.41		2533.05		3137.10		2242.61		2048.29	
p	0.000		0.000		0.000		0.000		0.000		0.000	

*** p<0.001, ** p<0.01, * p<0.05; Clustered standard errors in parentheses.

Essential and nonessential permission ratio are the key independent variables in columns (1) and (2)

Count of essential and nonessential permissions are the key independent variables in columns (3) and (4)

A dichotomous variable representing whether an app *overreached* is the key independent variables in columns (5) and (6)

A conclusive way to address this concern is to identify a sub-set of apps that already were seeking a-la-carte permissions (in some other app marketplace) during the period of study and study their upgrade decision on the Android Play Store. Apple iOS mandated a-la-carte permissions in 2012 for apps to even show up on any phone that had iOS 7. Thus, any Android app that has an equivalent iOS app should already have the required logic to seek a-la-carte permissions in place. We exploit this fact to study the upgrade decisions of this sub-group of apps. Of the 11,675 Android apps in our main analysis, 4,757 apps had an exact "twin" app on the Apple iOS App Store. We re-estimated model (1) on this sub-sample and present the results in Columns (1) and (2) of Table 4. We find that, even among apps that have an iOS equivalent, a 1% increase in *nonessential* permissions increases the propensity to delay upgrade by 0.8% ((1-0.992)*100=0.8%), ascertaining our finding.

4.2.1.2 Supply Side Motivation Reasons: The delay in upgrade may be driven by apps where developers lost interest in maintaining them. These could be older apps where developers gathered a lot of sensitive information but later found it a lot of work to re-wire the logic to minimize permission gathering. Such apps are likely to be abandoned, and hence, are less likely to be upgraded. A conclusive way to address this concern is to identify a sub-set of apps that are being actively maintained. Since 2018, Android has been following a strict de-listing policy for apps that are not actively upgraded (Android 2021). This means that any app that were on the Play Store at the end of 2019, 4 releases after Android 6.0 are the "actively maintained apps" (a total of 8,211 out of 11,675 apps). We re-estimated model (1) on this subgroup of apps. As seen in columns (3) and (4) of Table 4, even among apps that were actively maintained in 2019, a 1% increase in *nonessential* permissions increases the propensity to delay upgrade by 0.9% ((1-0.991)*100=0.9%), further ascertaining our main finding. In an unreported test, we re-estimated the model on apps that have been active as of end of 2021 (a total of 6,932 apps) and the outcomes are consistent.

4.2.1.3 Other Platform-Level Change in Android 6.0: Deliberate delaying behavior is typically introduced when platforms introduce new restrictions. Android 6.0 upgrade was very similar to Android 5.0 upgrade except that Android 6.0 introduced a new restriction on gathering sensitive information. Hence, a good falsification test is to check whether seeking *nonessential* permissions affect apps' decision to

upgrade to a version that did not restrict gathering *runtime/nonessential* permissions. To test this, we re-estimated model (1) by changing *failure* to a dichotomous variable that carries the value 1 when and after the app upgrades to Android 5.0 (the version that did not restrict runtime permissions). Columns (5) and (6) of Table 4 carries the results of the falsification test. Unlike in Table 3, seeking $nonessential_permissions_{it}$ do not affect apps' decision to delay upgrade[23] to Android 5.0. This evidence further confirms the argument that the decision to delay upgrade is strongly influenced by *overreaching* behavior.

4.2.2 Alternative measure of *essential/nonessential* permissions

As an alternative measure to Word2Vec based peer group, we use the IAB categorization, procured from a mobile app intelligence firm. As noted earlier, IAB (Interactive Advertising Board) categories are declared by apps for advertising accuracy and are much more granular and are accurate compared to the Android categories. There were a total of 120 IAB categories among the 13,691 apps. We use these 120 categories instead of the 411 sub-categories that we obtained from the Word2Vec exercise that we had performed in section 3.1. Once the app list for each of the 120 categories are obtained, we perform cutoff polling with 80% as the threshold, consistent with prior literature (e.g., Taylor and Martinovic 2016). We re-estimate model (1) using the new measure of *essential/nonessential* permissions. The results, presented in Appendix Table A4 is consistent with the main analysis.

Next, to test the sensitivity of the cutoff polling technique, we vary the cut-off from 70% to 90% to test the sensitivity of our cutoff polling method. As shown in Appendix Table D1, results are consistent with the main model. [24]

[23] It is worth noting that the HR for $dayssinceupdate_{it}$ is consistent with Table 3 (i.e., negative, and significant) suggesting that app developers' behavior wrt. maintenance cycles haven't changed much.

[24] Lowering the cut-off too low (e.g., 50%) would potentially result in misclassification of nonessential permissions as *essential* permissions, a scenario we wish to avoid.

Table 4. Robustness Checks

Dependent variable - $upgrade_{it}$	Model (R1)		Model (R1)		Model (R2)		Model (R2)		Model (R3)		Model (R3)	
Robustness Type	*Apps with matching iOS apps*		*Apps with matching iOS apps*		*Apps that are active in 2021*		*Apps that are active in 2021*		*Falsification: Upgrade to Android 5.0*		*Falsification: Upgrade to Android 5.0*	
	Cox PH Model		Mixed Effects Model		Cox PH Model		Mixed Effects Model		Cox PH Model		Mixed Effects Model	
$essential_permissions_{it}$	1.004*	(0.002)	1.066***	(0.004)	1.001	(0.002)	1.062***	(0.003)	0.015**	(0.005)	1.050***	(0.006)
$nonessential_permissions_{it}$	0.992***	(0.001)	0.978***	(0.002)	0.991***	(0.001)	0.979***	(0.001)	-0.001	(0.003)	0.995	(0.003)
$\Delta rating_count_{it}$	1.000	(0.000)	1.000***	(0.000)	1.000	(0.000)	1.000***	(0.000)	-0.000	(0.000)	1.000	(0.000)
$rating_{it}$	0.978	(0.031)	1.281**	(0.098)	0.960	(0.035)	0.957	(0.054)	-0.421***	(0.070)	0.539***	(0.058)
$log(popularity_{it})$	0.979	(0.011)	1.182***	(0.027)	0.986	(0.013)	1.244***	(0.022)	-0.017	(0.026)	1.156***	(0.037)
$games_i$	0.939	(0.038)	1.092	(0.101)	0.959	(0.045)	1.132	(0.076)	0.126	(0.104)	1.190	(0.175)
$personal_apps_i$	0.888**	(0.041)	1.250*	(0.137)	0.866**	(0.046)	0.992	(0.078)	-0.232	(0.128)	0.835	(0.156)
$utility_mobilespecific_i$	0.829***	(0.037)	0.743*	(0.090)	0.838***	(0.042)	0.753***	(0.057)	-0.105	(0.109)	0.868	(0.138)
$utility_mobileaccess_i$	0.756***	(0.039)	0.841	(0.093)	0.765***	(0.043)	0.835*	(0.072)	-0.401**	(0.140)	0.589*	(0.122)
$learn_explore_i$	0.962	(0.043)	0.954	(0.097)	0.954	(0.047)	1.026	(0.076)	-0.102	(0.128)	0.899	(0.164)
$category_i$= $content_consumption_i$ (baseline)												
$dayssinceupdate_{it}$	0.954***	(0.001)	0.958***	(0.002)	0.955***	(0.001)	0.957***	(0.001)	-0.060***	(0.001)	0.937***	(0.002)
$screenshots_{it}$	1.003	(0.002)	1.022***	(0.006)	1.005*	(0.003)	1.024***	(0.004)	-0.004	(0.006)	1.002	(0.009)
$log(filesize_{it})$	1.068***	(0.017)	1.139***	(0.043)	1.062***	(0.018)	1.137***	(0.029)	0.046	(0.036)	1.145**	(0.059)
$developer_appcount_{it}$	1.000	(0.000)	1.000	(0.001)	1.000	(0.000)	0.998***	(0.000)	0.000	(0.000)	0.999	(0.000)
ln(p)			171.031***	(2.262)			162.294***	(1.658)			140.198***	(3.696)
var(packageid)			183.130***	(44.164)			198.413***	(38.249)			23.847***	(9.379)
Total Count of Apps			4757				8211				3950	
Download Bucket Dummy	YES		YES		YES		YES		YES		YES	
Log Likelihood	-24094.18		-11663.72		-42079.34		-19411.78		-8735.29		-4790.95	
AIC	48224.36		23369.44		84194.69		38865.56		17506.57		9623.89	
BIC	48384.81		23556.63		84366.39		39065.88		17673.22		9818.31	
# of Obs.	54,937		54,937		102,631		102,631		77,522		77,522	
X^2	1299.36		1665.68		1808.90		2626.24		2255.27		1882.71	
p	0.000		0.000		0.000		0.000		0.000		0.000	

*** p<0.001, ** p<0.01, * p<0.05; Clustered standard errors in parentheses.

Subsample analysis for 4747 apps that have a "twin" on iOS is presented in columns (1) and (2)

Subsample analysis for 8211 apps that are actively maintained as of 2021 is presented in columns (3) and (4)

Subsample analysis for 3950 apps that upgrade from an earlier version of Android to Android 5.0 is presented in Columns (5) and (6)

4.2.3 Other alternative explanations

In Appendix D, we attempt to rule out a few other alternative explanations in cases where *overreaching* behavior are displayed primarily by apps that are feeder apps (e.g., Bank app, ticket booking app), controller apps (e.g., thermostat app) or personal category app (e.g., dating or social networking apps). The subsample analyses suggest that none of these reasons significantly account for the delay in upgrade.

In summary, our finding that apps that overreach are more likely to delay upgrading holds after accounting for multiple alternative scenarios.

4.3. Uncovering the mechanism: a key reason for these apps to delay upgrading?

In this section, we investigate one of the reasons why certain apps overreach. We start with investigating which specific runtime permissions are over-sought more frequently. We modify the model (1) to include a vector of permission dummies instead of an aggregate measure of permissions. The results, presented in Appendix Table A5, suggest that seeking specific permissions to access GPS location, phone status, access microphone, photos and phone memory, call logs as well as to send text messages, are more likely to predict a delay in the upgrade (i.e., HR<1 and significant)[25]. These very permissions are often listed among the most abused permissions relating to sensitive data gathering (e.g., Bankhead 2018, Weeks 2018, Kroger and Raschke 2019). Specifically, these are the permissions that have often been reported in media as associated with serving targeted advertising (Tiffany 2018, Limer 2018, Heddings 2018, Al-Hussaini 2020, Golubev 2019, Goode 2018).

4.3.1 Exploratory Text Mining Approach

To ascertain the intuition about the link between targeted advertising and overreaching behavior, we obtained a list of all media reports from January 2009 and October 2019 from Lexis Uni database. We search for reports with keywords "adware" and "Android" and found a total of 1559 media reports that were tagged as English articles. In these articles, the 6 permissions above were mentioned 3.2 times more

[25] HR for Precise Location is 0.864, this means that seeking permission for precise location increases the propensity to delay upgrade by 13.6% (1-0.864*100). Similarly, permissions for phone status, microphone, memory, call-log and sms predicts a delay in upgrade by 32.5%, 15.6%, 19.3%, 7.9% and 42.6% respectively.

than the rest of the 16 runtime permissions, suggesting a strong correlation between these permissions and serving intrusive advertisements. In Appendix B, we outline the use of machine learning techniques to further support our intuition that the decision to delay the upgrade is driven by apps' ad-targeting behavior. Specifically, we employ a combination of SkipGram based Topic Modeling and Lasso Logit to identify topics that predict delaying the upgrade. We find high similarities (cosine similarity) between these topics and media report clippings that report on aggressive advertising or "adware" in Android. It stands to reason then, that apps that rely on users' sensitive information to generate revenue (e.g., serving ads) are more likely to overreach permissions and therefore strategically delay the upgrade so that they continue gathering sensitive information.

4.3.2 Econometric Model

Here, we econometrically investigate whether app's advertising revenue model impacts its likelihood of delaying upgrading. Of the two main *freemium* revenue models in the Play Store, i.e., *in-app-advertising (IAA)* and *in-app-purchases (IAP)*, apps with an *IAA* revenue model to have stronger incentives to seek more *runtime* permissions and consequently delay upgrade. Alternatively, *IAP* revenue model should not impact apps' decision to delay the upgrade. Here, we modify the model (2) to include interaction terms between the permission values (*essential* and *nonessential* permissions) and the revenue models (*IAA* and *IAP*). The results are shown in Table 5. Columns (1) and (2) present outcomes of standard Cox PH model and Mixed Effect survival model, respectively.

In both model specifications, the HR of the interaction term between *nonessential* permissions and *IAA* revenue model is significant and less than 1, while the interaction between *essential* permissions and *IAA* is insignificant. Similarly, the interaction term between *IAP* and both types of permissions is insignificant. This finding empirically supports our intuition that having in-app-advertising as the revenue model further exacerbates the impact of *nonessential* permissions on apps' propensity to delay the upgrade. To visualize the findings, we plot the *essential* and *nonessential* permissions sought by apps across the Android versions over 3 different periods spread across our panel. As shown in Figure 6, there is a significant drop in seeking

nonessential permissions moving from *targetAPI* 22 (Android 5.1) to *targetAPI* 23 (Android 6.0), but insignificant changes in *essential* permissions, further adding strength to our arguments.

Table 5. Analysis of Upgrading to the Latest Version of Android: Interaction between Revenue Models and *essential* and *nonessential* Permissions

Dependent variable - $upgrade_{it}$	(1) *Hazard Ratios*		(2) *Hazard Ratios*	
Type of Model	Cox Proportional Hazard Model		Mixed Effects Survival Model	
$essential_permissions_{it}$	1.005*	(0.002)	0.996	(0.002)
$nonessential_permissions_{it}$	0.993***	(0.001)	0.993***	(0.002)
$inAppAdvertising_{it}$	1.548***	(0.171)	0.432***	(0.072)
$inAppPurchases_{it}$	1.087	(0.076)	1.621***	(0.189)
$\boldsymbol{essential_permissions_{it} X inAppAdvertising_{it}}$	0.993	(0.010)	0.990	(0.011)
$\boldsymbol{essential_permissions_{it} X inAppPurchases_{it}}$	0.987	(0.005)	0.998	(0.007)
$\boldsymbol{nonessential_permissions_{it} X inAppAdvertising_{it}}$	0.998**	(0.005)	0.960***	(0.008)
$\boldsymbol{nonessential_permissions_{it} X inAppPurchases_{it}}$	0.996	(0.003)	0.992	(0.005)
$\Delta rating_count_{it}$	1.000	(0.000)	1.000***	(0.000)
$rating_{it}$	0.980	(0.031)	0.981	(0.047)
$log(popularity_{it})$	0.977*	(0.011)	1.210***	(0.018)
$games_i$	0.939	(0.038)	0.938	(0.056)
$personal_apps_i$	0.892*	(0.041)	0.976	(0.065)
$utility_mobilespecific_i$	0.829***	(0.037)	0.738***	(0.049)
$utility_mobileaccess_i$	0.754***	(0.039)	0.775**	(0.062)
$learn_explore_i$	0.961	(0.043)	0.992	(0.067)
$category_i$=content_consumption$_i$ (baseline)				
$dayssinceupdate_{it}$	0.954***	(0.001)	0.956***	(0.001)
$screenshots_{it}$	1.003	(0.002)	1.022***	(0.004)
$log(filesize_{it})$	1.067***	(0.017)	1.155***	(0.027)
$developer_appcount_{it}$	1.000	(0.000)	0.998***	(0.000)
var(cons(packageid))			169.109***	(1.643)
Count of Apps	11,675		11,675	
Download Category Dummies	YES		YES	
Log Likelihood	-56277.62		-25269.81	
AIC	112603.2		50593.62	
BIC	112842.4		50862.64	
# of Obs.	156,903		156,903	
X^2	2293.33		2426.45	
P	0.000		0.000	

*** p<0.001, ** p<0.01, * p<0.05; Clustered standard errors in parentheses.

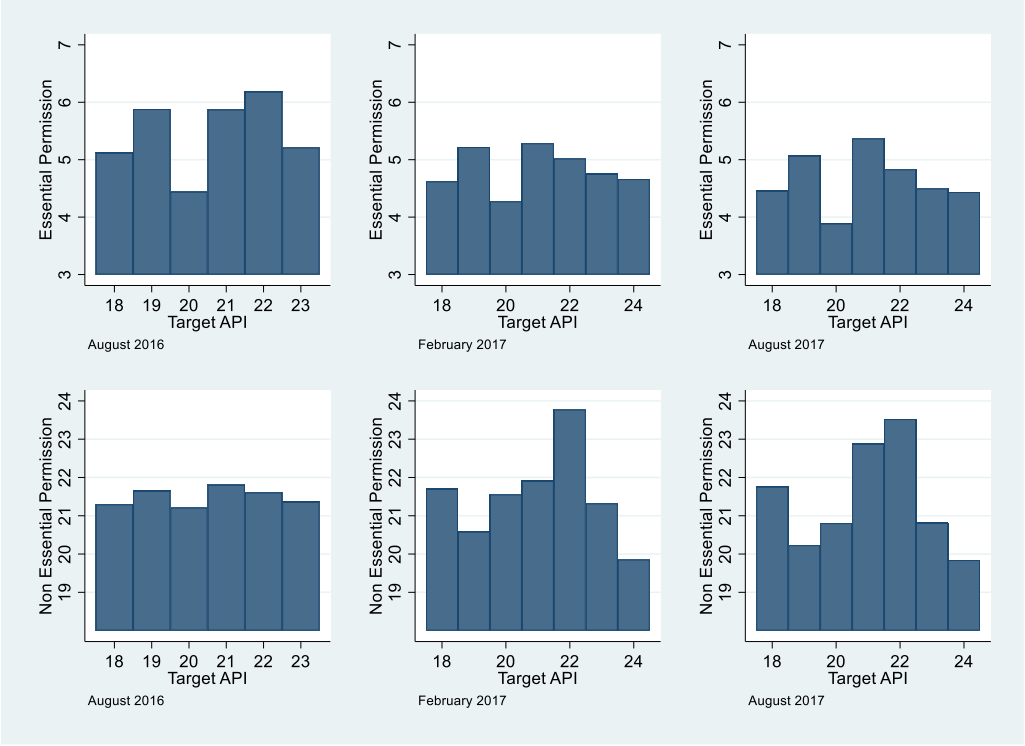


Figure 6: The average ratio of *essential* and *nonessential* permissions sought for each *targetAPI* as seen in three equally spaced time-periods (Aug 2016 through Aug 2017). API 20 is Watch OS version, hence the deviation.

4.4 RQ2: How does delaying upgrading by apps affect their marketplace outcomes (rating and popularity)?

Armed with the knowledge that some apps deliberately delay upgrading for advertising revenue reasons, we seek to understand if such delaying has any repercussions on their outcomes in terms of ratings and popularity. Empirically establishing and measuring such repercussions is helpful to both platform and developers that have the capabilities to upgrade on time but hold-off upgrading deliberately. To address the questions on the impact of delaying upgrading, we employ Propensity Score Matching (PSM) to preprocess the data (Dehejia and Wahba 2002) followed by Difference-in-Differences technique (DiD) technique to establish causal relationship between the treatment and the treated (Meyer 1995).

4.4.1 Model Setup

The timing of upgrading to the latest version is a choice that apps exercise, hence the action of upgrading is considered a *treatment* in this context. A DiD approach enables us to compare and contrast the effects on apps that upgrade to the latest versions (the *treated* apps), with those apps that do not upgrade until the end

of our panel (the *control* apps) [26]. Since the *treated* apps upgrade over a period of several months, i.e., in a staggered manner, we set the month of treatment (month when the *treated* app upgrades) to time 0 and adjust the months pre- and post- that in reverse chronological order (…-3, -2, -1) and sequential order (1, 2, 3…), respectively. Such a normalization of time is performed based on prior research (Autor 2003, Fang et al. 2014, Gao and Zhang 2016). Employing DiD estimation approach in a nonexperimental setup such as ours requires that we correct for treatment selection bias. To address this, we employ (1) PSM as our main technique and (2) Look-Ahead PSM (LA-PSM) to ascertain the robustness of matching procedure. PSM has been used in many nonexperimental settings (e.g., Dehejia and Wahba 2002). Specifically, techniques combining PSM to preprocess the dataset, followed by DiD have been used in research where the treatment may be affected by selection bias (Smith and Todd 2005, Liu and Lynch 2011, Mayya et al. 2021). LA-PSM is a newer technique used by researchers to address similar biases (Bapna and Umyarov 2015). Both techniques match *treated* samples with *control* samples based on a propensity score that is generated using a combination of time-variant and invariant covariates. If matched correctly, the *treated* and *control* samples will have the same propensity score to upgrade in a given month (based on covariates), with the decision to upgrade being the only difference. In other words, each *treated* app (an app that upgrades within the time-period of the panel) is matched with a *control* app that never upgrades until the end of the panel based on the propensity scores. We apply nearest one-to-one neighbor matching without replacement of the *control* samples. To account for the difference in time periods when treatments are introduced (app A upgrades in June 2016, app B upgrades in July 2016), we follow a dynamic matching technique where we match the *treated* apps with *control* apps one month before the month of treatment.

A major distinction between the PSM and LA-PSM is the *control* sample. In the PSM technique, each *treated* app is matched with a *control* app that never upgrades until the end of the panel. On the contrary, LA-PSM performs matching not only on time-variant/invariant covariates but also on the intent to switch. Specifically, in LA-PSM, control samples are drawn from a pool of apps that will eventually upgrade

[26] Later on, we will explain how we further bifurcate treated apps into apps that delayed upgrade vs those that did not.

sometime in the future but not during the month under consideration. A key thing to note about LA-PSM is that, compared to PSM, a larger proportion of *treated* samples stay unmatched and are consequently dropped from the analysis because the pool of *control* apps shrinks as we reach the end of the panel.

The covariates used for matching include (a) time variant app characteristics such as the count of rating, file size and date since the latest update, and (b) key variables of interest such as *essential* permissions, *nonessential* permissions, app popularity and app rating. Also, we explicitly ensure that *treated* and *control* apps come from the same download bucket, belong to the same app category and have very similar upgrade frequency (i.e., *dayssincelastupdate*). The PSM procedure gives us 2504 *treated* and 2504 *control* mobile apps in the final sample, while the LA-PSM procedure gives us 1347 *treated* and 1347 *control* mobile apps in the final sample.

Despite the pre-processing and matching as described above, a quasi-experimental setup such as ours may suffer from simultaneity in the determination of the two key variables, *rating* and *popularity*. Given that we allow for endogenous variables $rating_{it}$ and $popularity_{it}$ to enter the right-hand side of the other equation, the error terms of the equations may be correlated with one another. To address this concern, we estimate the two equations simultaneously by employing the three-stage least squares (3SLS) method. Such an approach is common in IS literature (e.g., Aral et al. 2018; Chang and Gurbaxani 2013; Mayya et al. 2021). For the two equations to be identified, we use the first difference of the dependent variables as the respective excluded instruments in the equation.

Table 6. Covariate Balance – PSM Technique

Covariates	Pre-Match (Overall)	Pre-Match (Treated)	Pre-Match (Control)	Difference	t-test	Post-Match (Treated)	Post-Match (Control)	Difference	t-test
Rating Count	48410.14	52772.67	46468.36	-6304.31	-2.58	25203.86	26952.3	1748.44	0.89
File Size	22.70	23.34	22.46	-0.88	-6.35	26.01	25.98	-0.03	-0.04
dayssinceupdate	189.65	108.17	219.05	110.88	99.71	114.02	117.87	3.84	1.15
developer_appcount	30.93	25.88	32.76	6.87	12.84	27.56	29.88	2.32	1.09
runtime_permissions	27.00	27.25	26.91	-0.34	-4.37	27.56	27.82	0.26	0.74
essential_permissions	3.93	3.12	4.22	034	32.69	2.84	2.83	-0.01	-0.03
nonessential_permissions	23.07	24.13	22.69	-1.48	-19.24	24.72	24.98	0.26	0.76
rating	3.99	4.02	3.98	-0.04	-17.70	3.98	3.99	0.01	1.37
popularity	$6.31x10^{\wedge-6}$	$7.20x10^{\wedge-6}$	$5.98x10^{\wedge-6}$	$-1.22x10^{\wedge-6}$	-21.95	$5.31x10^{\wedge-6}$	$5.53x10^{\wedge-6}$	$0.23x10^{\wedge-6}$	1.56

Note: Apps that were already in Android 6.0 at the start of the panel are excluded from PSM DiD analysis.

As the first test for successful matching procedure, we compare continuous covariates in our models before and after the matching procedure. The outcomes of covariate balance tests for the PSM technique is presented in Table 6 (and that for LA-PSM in Panel B of Appendix Table A6). The covariates are well balanced after matching.

4.4.2 Determining if apps are delaying upgrading to Android 6.0

Apps indulging in deliberate delaying behavior is beneficial to neither the customers (they lose out on enhanced security, improved reliability, better UI and UX) nor the platform (because it would exacerbate the existing issue of supporting multiple SDK versions). Hence, it is useful to investigate if such delaying has repercussions on apps' outcomes. To study the effect of deliberate delaying on apps' marketplace outcomes, we distinguish early/on-time upgraders from late upgraders among *treated* apps in our models. Since Android launches a new version every year, we consider all those apps that upgraded to Version 6.0 (or above) after users started to upgrade to Version 7.0 of Android as late upgraders among apps. While Android announced Version 7.0 at the end of August 2016, 0.1% of overall Android customers had installed the update by November 2016. Hence, we consider all apps that upgraded from version 5.1 or below (blanket permission mechanism) to 6.0 or a higher version (runtime permission mechanism) in and after November 2016 as late upgraders.

Econometrically, we estimate the impact of delaying upgrade on apps' marketplace outcomes as follows:

$$\boldsymbol{rating_{it}} = \beta_0 + \beta_1 * upgrade_group_i * post_upgrade_t + \beta_2 * late_upgrade_group_i * post_upgrade_t + \beta_3 * log(popularity_{it}) + \beta_4 * \Delta rating_count_{it} + \beta_5 * runtime_permissions_ratio_{it} + \beta_6 * dayssinceupdate_{it} + \beta_7 * log(filesize_{it}) + \beta_8 * screenshots_{it} + \beta_9 * developer_appcount_{it} + \beta_{10} * download_bucket_i + \beta_{11} * month_dummy_t + w_i + \eta_{it} \quad (2a)$$

$$\boldsymbol{log(popularity_{it})} = \alpha_0 + \alpha_1 * upgrade_group_i * post_upgrade_t + \alpha_2 * late_upgrade_group_i * post_upgrade_t + \alpha_3 * rating_{it} + \alpha_4 * \Delta rating_count_{it} + \alpha_5 * runtime_permissions_ratio_{it} + \alpha_6 * dayssinceupdate_{it} + \alpha_7 * log(filesize_{it}) + \alpha_8 * screenshots_{it} + \alpha_9 * developer_appcount_{it} + \alpha_{10} * download_bucket_i + \alpha_{11} * month_dummy_t + w_i + \eta_{ait} \quad (2b)$$

In the above models, $upgrade_group_i$ is a binary variable that carries a value of 1 for apps in the *treated* group and 0 for apps in the *control* group. Similarly, $late_upgrade_group_i$ is a binary variable that carries

the value of 1 for treated apps that delay upgrading and 0 for the rest of the apps. For each treated-control pair, the variable $post_upgrade_t$ carries a value of 0 before the *treated* app performs an upgrade and 1 after it upgrades to the latest version. Since we follow the time normalization procedure, $late_upgrade_group_i * post_upgrade_t$ identifies the effect of delaying upgrading, over and above the effect of upgrading (captured by the interaction term $upgrade_group_i * post_upgrade_t$**).** This coefficient equals 1 only for treated apps that delayed upgrade ($late_upgrade_group_i$ =1) post upgrade ($post_upgrade_t$=1). Coefficients α_2 and β_2 in the above models above provide us with required DiD estimates (impact of delaying upgrade). The variable w_i represents the app-level fixed effects to account for any app level unobserved heterogeneity. Finally, due to the skewed nature of the variable $popularity_{it}$, it has been log transformed in both equations.

4.4.3 Parallel Trends

Another important test for quasi-experimental setup is the Parallel Trends assumption. The assumption states that the *control* and *treated* apps should have a similar time trend before the treatment. To test this assumption, we follow the recommendations in literature (Autor 2003) where we model time trends of the dependent variables in the DiD models. This can be done by introducing interaction terms of the treatment and various month dummies. The models that we use for testing the parallel trends assumptions are as follows:

$$\begin{aligned} \boldsymbol{rating_{it}} = {} & \beta_0 + \sum \beta_j^p * upgrade_group_i * relative_month_j + \beta_1 * \Delta rating_count_{it} + \beta_2 * \\ & log(popularity_{it}) + \beta_3 * runtime_permissions_{it} + \beta_4 * dayssinceupdate_{it} + \beta_5 * log(filesize_{it}) + \beta_6 * \\ & screenshots_{it} + \beta_7 * developer_appcount_{it} + \beta_8 * download_bucket_i + \beta_9 * month_dummy_t + w_i + \eta_{it} \end{aligned} \quad (3a)$$

$$\begin{aligned} \boldsymbol{log(popularity_{it})} = {} & \alpha_0 + \sum \alpha_j^p * upgrade_group_i * relative_month_j + \alpha_1 * \Delta rating_count_{it} + \alpha_2 * rating_{it} \\ & + \alpha_3 * runtime_permissions_{it} + \alpha_4 * dayssinceupdate_{it} + \alpha_5 * log(filesize_{it}) + \alpha_6 * screenshots_{it} + \alpha_7 * \\ & developer_appcount_{it} + \alpha_8 * download_bucket_i + \alpha_9 * month_dummy_t + w_i + \eta_{ait} \end{aligned} \quad (3b)$$

where, the variables $relative_month_j$ is a vector of dummies for each month *j* before the treatment occurred. If the point estimate of the interaction terms (i.e., α_j^p and β_j^p) are insignificant before the treatment, the parallel-trends assumption holds. We list the outcome of these models in Panel A of Appendix

Table A6. All interaction terms before the month of upgrade are insignificant, therein assuring that the Parallel Trends Assumption is not violated by the matching technique. Figure 7 provides the pictorial representation of parallel trends assumption for *rating* and *popularity* as well as the variables, *essential* and *nonessential* permissions. Panel C of Appendix Table A6 shows that even the LA-PSM model does not violate the parallel-trends assumption.

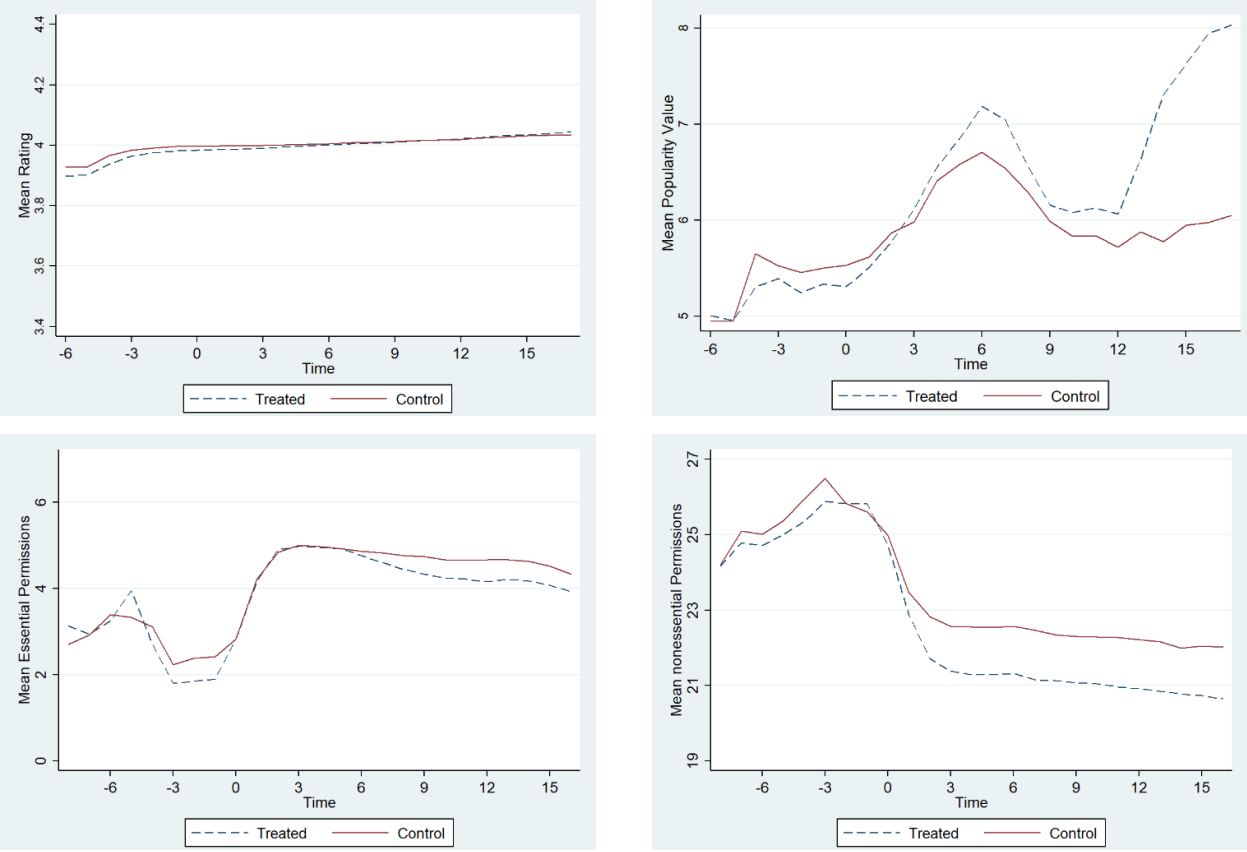


Figure 7: Graph showing the parallel-trends assumption for treated and control apps for rating, popularity, *essential* and *nonessential* permissions sought for each Target API. Time 0 is the month of matching.

4.4.4 Estimation Results

The outcomes of the DiD estimation are shown in Columns (1) and (2) of Table 7. Columns (1) and (2) present the standard DiD estimation while columns (3) and (4) present the estimation by including an interaction term between $late_upgrade_group_i$ and $post_upgrade_i$. This interaction term identifies the effect of delaying the upgrade on the dependent variables, over and above the effect of upgrading to the latest version of the app (which is captured by the interaction term between $upgrade_group_i$ and $post_upgrade_t$).

Column 3 shows the estimate of model 2(a). Here, we find that the rating of apps that delay upgrade see a drop in rating by 0.223% (-0.009/4.04 average rating = - 0.223%). For an average app in the dataset, this drop is an equivalent of 1267 app users reducing the rating by 1 star (i.e., 140776 average rating * 0.009 = 1267 overall star decrease). Comparing this estimate with the coefficient of the interaction term between $upgrade_group_i$ and $post_upgrade_t$, delaying the upgrade almost completely erases the positive bump in the rating that apps would have received upon upgrading to the latest version on time.

Column 4 of Table 7 shows the results for model 2(b). Here we find that the apps suffer a large loss in their popularity, measured by PageRank. The point estimate -0.251 suggests that delaying the upgrade reduces the popularity of the app by 25.2% (i.e., exp(-0.290) = 0.748 or (1-0.748)*100 = 25.2% drop). This is a massive drop in the popularity in the app. Because app downloads are extremely skewed toward the highest ranked app even among the top apps (Garg and Telang 2013), a 25.2% drop in popularity would result in a massive drop in downloads.

Table 7. Analysis of Effects of Delaying Upgrading to Latest Version

Dependent Variable	(1) DID – $rating_{it}$		(2) DID – log($popularity_{it}$)		(3) DID – $rating_{it}$		(4) DID – log($popularity_{it}$)	
Model	3SLS		3SLS		3SLS		3SLS	
upgrade_group$_i$*post_upgrade$_t$	0.007***	(0.000)	0.078**	(0.025)	0.009***	(0.001)	0.156***	(0.034)
late_upgrade_group$_i$*post_upgrade$_t$					**-0.009*** **	**(0.001)**	**-0.290*** **	**(0.038)**
log(popularity$_{it}$)	0.002***	(0.000)			0.002***	(0.000)		
rating$_{it}$			-12.936***	(3.544)			-12.929***	(3.539)
Δrating_count$_{it}$	0.000***	(0.000)	0.000***	(0.000)	0.000***	(0.000)	0.000***	(0.000)
runtime_permissions$_{it}$	0.000***	(0.000)	0.003***	(0.001)	0.000***	(0.000)	0.003***	(0.001)
dayssinceupdate$_{it}$	-0.000***	(0.000)	-0.001***	(0.000)	-0.000***	(0.000)	-0.001***	(0.000)
screenshots$_{it}$	0.001*	(0.000)	0.007	(0.010)	0.001*	(0.000)	0.007	(0.010)
log(filesize$_{it}$)	0.002***	(0.001)	0.106***	(0.015)	0.002**	(0.001)	0.101***	(0.015)
developer_appcount$_{it}$	0.000***	(0.000)	0.000	(0.000)	0.000***	(0.000)	0.000	(0.000)
Constant	4.232***	(0.010)	43.242**	(14.923)	4.232***	(0.010)	43.208**	(14.902)
Count of Apps	5008 (2504 treated and 2504 control)				5008 (2504 treated and 2504 control)			
App Fixed Effect	YES		YES		YES		YES	
Month Dummies	YES		YES		YES		YES	
Download Bucket Dummies	YES		YES		YES		YES	
# of Obs.	114079		114079		114079		114079	
R-Squared	0.991		0.517		0. 991		0.518	

*** p<0.001, ** p<0.01, * p<0.05;

4.4.5 Validation Test: Change in App's Permission Seeking Behavior

RQ1 argued that overreaching apps are more likely to delay the upgrade. Because of overreaching, these delaying apps have to re-design their permission seeking procedure before upgrading, otherwise they risk

attracting public scrutiny of their overreaching behavior. Thus, we expect the apps to significantly reduce seeking *nonessential* permissions upon upgrade, much more than apps that upgrade on-time. We test this by estimating models similar to 2(a) and 2(b), but by replacing the dependent variables with $essential_permissions_{it}$ and $non_essential_permissions_{it}$ respectively. Consistent with the earlier analysis, we estimate the two results simultaneously and allow the variables to enter the right-hand side of the other equation. Also, consistent with our earlier analysis, we use the first difference of the dependent variable as the excluded instrument for the system of equations to be identified. Table 8 shows the results of the estimation.

Table 8. Delaying Upgrade and Apps' Permissions Seeking Behavior - PSM

Dependent Variable	(1) DID - $essential_permissions_{it}$		(2) DID – $nonessential_permissions_{it}$		(3) DID - $essential_permissions_{it}$		(4) DID – $nonessential_permissions_{it}$	
Model	3SLS		3SLS		3SLS		3SLS	
$\boldsymbol{upgrade_group_i * post_upgrade_t}$	0.050	(0.037)	-0.953***	(0.072)	-0.061	(0.042)	-0.832***	(0.081)
$\boldsymbol{late_upgrade_group_i * post_upgrade_t}$					0.414***	(0.071)	-0.450**	(0.141)
$essential_permissions_{it}$			0.451***	(0.024)			0.451***	(0.024)
$nonessential_permissions_{it}$	-0.082***	(0.006)			-0.082***	(0.006)		
$rating_{it}$	0.448	(0.238)	1.017*	(0.469)	0.491*	(0.238)	0.970*	(0.469)
$\Delta rating_count_{it}$	0.000***	(0.000)	-0.000***	(0.000)	0.000***	(0.000)	-0.000***	(0.000)
$dayssinceupdate_{it}$	0.000***	(0.000)	0.004***	(0.000)	0.000***	(0.000)	0.004***	(0.000)
$screenshots_{it}$	0.412***	(0.033)	-0.644***	(0.066)	0.413***	(0.033)	-0.644***	(0.066)
$log(filesize_{it})$	-0.022	(0.049)	0.314***	(0.096)	-0.016	(0.049)	0.308**	(0.096)
$developer_appcount_{it}$	0.005***	(0.001)	0.000	(0.001)	0.005***	(0.001)	0.001	(0.001)
Constant	-4.993***	(1.236)	36.162***	(2.407)	-5.164***	(1.236)	36.351***	(2.408)
Count of Apps	5008 (2504 treated and 2504 control)				5008 (2504 treated and 2504 control)			
App Fixed Effect	YES		YES		YES		YES	
Month Dummies	YES		YES		YES		YES	
Download Bucket Dummies	YES		YES		YES		YES	
# of Obs.	114079		114079		114079		114079	
R-Squared	0.777		0.812		0. 777		0.812	

*** p<0.001, ** p<0.01, * p<0.05;

From column (4) of table 8, we see that apps reduce seeking *nonessential* permissions by 0.832 units (or 0.832 percent, since the dependent variable is out of 100) upon upgrading (interaction term between $upgrade_group_i$ and $post_upgrade_t$). What's interesting is that the apps that delay upgrade further reduce seeking *nonessential* permissions (interaction term between $late_upgrade_group_i$ and $post_upgrade_t$) by 0.450 units (or 0.450%,). However, we don't see a corresponding reduction in *essential* permissions upon

delaying the upgrade. Overall, this finding supports the intuition that overreaching apps delay the upgrade [27]

4.5 Robustness Checks

4.5.1 Look-Ahead Propensity Score Matching

For the first robustness check, we re-estimated our models by employing LA-PSM instead of PSM model. This is to ascertain that the outcomes that we observe are not the artifacts of matching procedure. As noted earlier, the control group of apps comes from the pool of apps that upgrade to version 6.0 or higher at a later time period. Since the "control" apps in LA-PSM analysis are apps that delay upgrade, we don't have to include a 3-way interaction term with the "late upgrader" dummy. The results are presented in Appendix Table A7. Per the LA-PSM model, delaying upgrade costs an equivalent of about 563 viewers reducing the rating by 1 star (DiD estimate for upgrading on time is 0.004. Therefore, apps that delay upgrade lose out on a rating bump of 0.004*140776 average rating = 563.10 overall stars). Similarly, the models suggest that delaying results in a loss of popularity for those apps by 5.65% (DiD estimate for upgrading on time is 0.055, exp(0.055) = 1.0565 or 5.65% positive bump in popularity upon upgrading on time). Both the results are numerically and statistically consistent with our main model. The lower magnitude of the effects is expected since, unlike the PSM models, the control apps in LA-PSM model upgrade after a few months. In summary, the outcomes of Look-Ahead Propensity Score Matching analyses are qualitatively and quantitatively very similar to PSM+DiD analyses.

4.5.2 Addressing Serial Correlation

Next, to ensure that our DiD estimates are robust to a potential serial correlation in the residuals, we followed two recommendations prescribed in the literature. The first recommendation is from Bertrand et

[27] These above results are interpreted with caution. Despite a DiD model, we do not claim that the results of table 8 are causal in nature (i.e., delaying upgrade reduces nonessential permissions). The interpretation here is that apps that delay upgrade are the ones that potentially overreach and therefore are the ones that must correct this overreaching while upgrading. These happen simultaneously (decision to upgrade and decision to reduce overreaching) and not necessarily causal. The overall goal of these estimations is to provide strong support to our core argument that overreaching behavior triggers delay in upgrade.

al. (2004). We collapsed and averaged the time series data into a panel of length 2—a *pre-treatment* time point and a *post-treatment* time point—which circumvents the autocorrelation issue without making any specific assumption about its form. This resulted in a total of 10016 observations, 5008 *pretreatment* and 5008 *post-treatment* observations. The results are shown in Appendix Table A8. We observed a negative and significant coefficient for $late_upgrade_group_i * post_upgrade_t$ in both the equations for rating and popularity (Panel A of Appendix Table A8). Specifically, Panel A of Appendix Table A8 suggests that delaying upgrading results in a rating drop, equivalent to about 2112 viewers reducing the rating by 1 star (i.e., 140776 average rating * 0.016 = 2111.64 overall star decrease). Similarly, column (2) of the table shows that delaying the upgrade reduces the popularity of the apps by 38.06% (i.e., exp(-0.479) = 0.6194 or (1-0.6194)*100 = 38.06% drop). It is worth noting that the coefficients in these models are larger than the ones in our main model. More importantly, this finding suggests that the PSM+DiD models are not severely affected by serial correlation issues. We find qualitatively consistent results with LA-PSM + DiD models (Panel B of Appendix Table A8).

The second recommendation that we follow is to bootstrap the standard errors, consistent with literature on Propensity Score Matching without replacement (Austin and Small 2014, Smith and Todd 2005). We estimate models 2(a) and 2(b) independently and then bootstrap the standard errors based on 500 replications. The results are presented in Appendix Table A9. The results are consistent with the main model. Both these analyses lend support to our claim that potential serial correlation problems will not alter the outcomes in our models.

4.5.3 Oversampling of Successful Apps

Our primary data collection process involved installing apps on emulators. We have taken care to ensure that the apps are representative of the global app population. Our data collection strategy involved utilizing a large sample of apps to ensure this representativeness, as outlined in detail in Appendix C. In Appendix Table C1, we demonstrate that the app distribution across categories in our sample closely resembles the overall app distribution across categories. Yet, questions regarding the generalizability of our findings may still exist. For example, the negative consequence for rating and popularity may be disproportionately

towards the more popular apps. Appendix C has a detailed discussion on the external validity of our finding. Here, we demonstrate that the impact of delaying the upgrade on marketplace outcomes such as apps' rating and popularity are conservative for the most popular apps. Table C2 in Appendix C shows the outcome of the analysis. Among the apps in the DiD model, the negative impacts of delaying are the lowest for the most popular apps, i.e., apps with over 50 million downloads. We find that the negative effect size becomes larger for less popular apps within our dataset and the largest for apps with less than five million downloads. This suggests that studying the negative consequences of delaying the upgrade on a sample of very popular apps is likely to produce a conservative estimate of the outcomes and that the repercussions are more severe for less popular apps.

4.5.4 Impact on Apps that do not reduce overreaching

It is useful to test the mechanism through which apps that delay upgrade get penalized. The key motivation for this platform policy change is to encourage users to carefully evaluate the permissions sought by apps and help them granularly decide each permission. The consequence of such a policy change is that users are expected to exercise their discretion and avoid installing annoying apps that do not reduce their overreaching behavior. Such annoyance or app avoidance should only increase with time, as they get comfortable denying permissions. Hence, we expect the negative impact on apps that upgrade but do not reduce overreach to be larger than those apps that reduce overreaching upon upgrading. To test this mechanism, we create a dichotomous variable, $continue_overreaching_i$ which carries the value of 1 if an app i continues asking the same or greater $nonessential\ permissions_{it}$ after upgrade and a value of 0 if an app i reduces overreaching. We interact this term with DiD term $late_upgrade_group_i * post_upgrade_t$. The results are presented in Appendix Table A10. We find that the negative impact of delaying the upgrade on both *rating* and *popularity* is more severe for apps that continue overreaching, as compared to apps that reduce their overreaching. This analysis provides additional support to our premise that the policy change negatively affects marketplace outcomes for overreaching apps via that policy's influence on user behavior.

5. Discussion and Conclusion

5.1 Discussion of the key findings

Our study is motivated by an increasing user demand to take control over accessing sensitive personal information and a dearth of research examining the impact of platforms' mechanisms that meet those demands. This research investigates apps' responses to Android's policy change that impacted how Android users granted permissions to apps. Specifically, the research studies apps' upgrade behavior and the consequent negative effects of such behavior on their own marketplace outcomes. In investigating the characteristics of apps that delay upgrade, we find that the propensity to delay upgrade increases for apps that have a sporadic maintenance cycle (based on the days since last update) or apps that provide minimal/specific functionalities (based on smaller file size). The finding is intuitive, since app maintenance cycles or functionalities are indicative of apps' demand or developers' resources. Next, we find apps from categories that provide personalized content such as personal (e.g., dating/social networking), mobile-specific (e.g., navigation, food delivery) or mobile access (e.g., banking, thermostat) are more likely to delay the upgrade. These finding counter prior work that show that apps in these categories (e.g., social, tools, personal, navigation) push out app updates much more frequently than games or content consumption apps (Mcllroy et al. 2016). Therefore, despite being known to send out frequent mini-updates and bug fixes, apps in these categories are more likely to delay upgrading, pointing towards a desire to continue harvesting user data.

Focusing on main explanatory variables, we find that apps that seek more *nonessential* permissions from users are more likely to delay upgrading to the latest version of Android. In other words, apps' overreaching behavior has a strong influence on apps' upgrade decision. The propensity to delay the upgrade increases when apps seek permissions that are silent (i.e., running in the background) or are known to be used for advertising personalization. Specifically, permissions to capture microphone audio, location, call logs or SMS, user identity or phone content have been utilized by ad networks to serve targeted advertisements (Tiffany 2018, Limer 2018, Heddings 2018, Al-Hussaini 2020, Golubev 2019, Goode 2018). Hence, those apps that typically benefit from in-app advertising prefer to retain control over such

permissions by delay upgrading. This is despite the documented penalty (on quality index) imposed by the platform for delaying. Clearly, apps believe that continuing to overreach has more gains than losses.

Next, our related analysis on *why apps overreach* (e.g., text mining analyses) suggests that apps' business models have something to do with such strategic delaying decision. Displaying *in-app-advertisements* is quite a common revenue model among free apps (Watanabe et al. 2017). Since targeted advertising is known to be effective (Farahat and Bailey 2012), it is understandable why apps that collect sensitive information for targeted advertising would choose to hold off upgrading to the restrictive platform version as much as they can. The ethics of gathering sensitive information just to serve targeted advertising is questionable. Especially, some ad-libraries that apps utilize are known to follow nonstandard data policies such as uploading users' sensitive information to remote servers or triggering code from remote servers on apps (Cimpanu 2018, Grace et al. 2012). Such reckless practices by advertisement networks (prone to security and privacy vulnerabilities) are not completely without the apps' knowledge (Grace et al. 2012). Given such questionable data gathering policies, apps would find it difficult to explain why certain sensitive information is required, which would result in delays. Relatedly, our finding is also consistent with prior research that suggests that obtaining consent to access sensitive information that is relevant to transactions (viz. *essential* permissions) is not difficult (Zimmer et al. 2010).

The main highlight of this research is to show that deliberately delaying the upgrade has strong negative consequences on apps' marketplace outcomes. First, we find that delaying the upgrade results in a drop in apps' rating. Delaying upgrade almost completely erases the boost in rating that apps would have seen upon upgrading on-time. Specifically, delaying upgrade results in an equivalent of 1267 users downvoting the app by 1 star. Given that online reputation system is skewed toward high rating (Reznick and Zeckhauser 2002), thousands of users giving lower rating can really affect the ranking of the app on the Play Store. Similarly, delaying the upgrade has a negative impact on apps' popularity score on the Play Store, with the PageRank falling by about 25.2%. Prior research on Android's platform has noted that highly ranked apps get significantly more future downloads than lower ranked ones (Garg and Telang 2013). Since free apps strongly rely on future downloads and new user engagement for continued source of revenue, a fall in

popularity or rating is going to affect their future downloads and, therefore, revenues. More importantly, this research quantitatively establishes the cost of delaying upgrade. By deciding when to upgrade, apps trade-off their short-term benefits (retaining control over users' information for a bit longer) with medium- to long-term costs of delaying the upgrade. These findings are quite important for open-hardware platforms such as Android. Even if an open platform cannot coerce all apps to upgrade immediately, our findings show that apps exploiting such flexibility for short-term gains come at a cost.

Next, in examining how delay upgrading to Android 6.0 is associated with apps' permission seeking, we find that apps delaying behavior is significantly associated with reducing *overreaching*, much higher than apps that upgrade on-time. There are two ways to interpret this finding. The first interpretation pertains to the correlation between overreaching and delaying upgrade. Apps that delay upgrade are the ones that overreach in the first place. Therefore, to avoid public scrutiny of their data gathering behavior, they reduce seeking *nonessential* permissions in a larger magnitude compared to apps that upgrade on time. The second interpretation pertains to how users' desire to control sensitive information strengthens over time. Giving fine-grained control to users over their private information improves apps' data gathering behavior in the short term and over time. Both interpretations point towards welfare improvement on the platform upon enacting this new policy change.

Finally, the heterogeneity analyses reveal many interesting insights. First, the auxiliary analyses in the appendix (Appendix Table A10) suggests that the negative impact of delaying the upgrade is significantly larger for apps that do not reduce overreaching. These findings suggest that app users are indeed sensitive to over-seeking behavior [28] so it is in the best interest of apps to reduce overreaching upon upgrading to the new privacy regime. Next, analyses also reveal that the negative impact of delaying is limited on highly popular apps and that the negative consequence of delaying upgrading increases in magnitude for apps with lower than five million downloads compared to apps with greater than 50 million downloads. While there

[28] Unlike feature improvements which individual apps need to opt-in to, privacy control improvements are applicable to all apps. Thus, while the existing app users won't be unhappy if apps delay feature enhancements, users are likely to be unhappy if they notice that apps are delaying privacy control (a list of all permissions is displayed each time users update the app).

are very few "super popular" apps that may bulldoze its way through overreaching, this is not the case with less popular apps. This result also helps establish the external validity of our finding. Table 9 below summarizes the findings of this research.

Table 9. Summary of Major Findings

Research Questions	Major Findings
RQ 1: What are the characteristics of apps that delay upgrading to seeking runtime permissions?	○ Delaying upgrading is likely for apps that overreach ○ A 1 percent or a 1-unit count increase in nonessential runtime permissions increases the propensity to delay upgrade by 0.8% or 5.1%, respectively ○ Apps that overreach, in general, are predicted to delay upgrade by 24.6% ○ Delaying upgrading are also predicted by apps' *in-app-ads* revenue model ○ Seeking permissions for sensitive information such as GPS, Call Logs, Microphone, Phone Status and Phone memory significantly predict a delay in upgrading. ○ The above-mentioned sensitive information is known to be over-sought by apps to improve targeted advertising within the app. ○ Thus, apps that rely on in-app-advertising have a higher incentive to retain blanket access to sensitive data. Hence, they delay upgrading.
RQ 2: How does delaying upgrading by apps affect their marketplace outcomes?	○ Apps' rating falls by 0.009, an equivalent of 1267 users downvoting the rating by 1 star. ○ Apps' popularity score (Weighted PageRank) falls by 25.2% upon delaying upgrading. ○ Apps with lower download counts are more severely impacted. ○ Apps that do not reduce their overreaching are more severely impacted.

5.2 Limitations and Future Research

We acknowledge several limitations of this study. First, similar to prior research on mobile app marketplaces (Garg and Telang 2013, Ghose and Han 2014), our sample focuses on the most popular apps at a given time (i.e., April 2016). The apps had to be installed on an emulator to detect when they upgraded, which precluded the use of hundreds of thousands of apps in our analysis. While we have put efforts to make our sample as representative as possible and conduct robustness checks that suggest our estimates are conservative for popular apps, generalizability concerns limit the findings of this study to overall apps, particularly those at the tail-end of the download distribution. Furthermore, our sample is limited to those apps that can be downloaded on emulators from the US Play Store. We need additional studies across a larger sub-sample of apps that would include apps with very number of low downloads as well as those that are only available in international markets.

Second, we rely on statistical techniques such as peer group analysis (Peddinti et al. 2019) or permission prevalence analysis (Taylor and Martinovic 2016), combined with textual analysis, to detect if an app is overreaching. This is a well-established method, since even Android uses Peer Group Analysis to

detect apps that overreach (Pelikan et al. 2017). However, such statistical techniques can result in misclassification. Despite an alternative measure of classification based on IAB categorization, we need more research to measure the essentiality of permissions in fundamentally different ways. Future research can interview app users to understand essentiality from their perspective. For example, one could derive an importance measure of individual apps to individual users. Whether and how upgrade decisions vary with an app's relative importance to users would provide additional insights.

Third, while our study shows overreaching behavior negatively affects users' opinion about overreaching apps (e.g., negative impact on rating), our study focuses on determining the average effect across all apps. There may be heterogeneity in the outcomes, depending on the app design and flow. For example, we do not observe in what level of app onboarding do these apps seek such *runtime* permissions. There may be some apps that are successful in overreaching by "educating" app users about the necessity of such permissions for the apps' survival. Future research can study app interface designs, specifically the messaging design that may convey to users why certain permissions are necessary to an app's working, thereby avoiding being denied.

Finally, despite using two distinct matching methods, DiD and simultaneous estimation of equations in a quasi-experimental setting, it is plausible that the self-selected decisions to delay the upgrade may be influenced by other unobservable factors. We need more research to leverage exogenous shocks (e.g., platform-level shocks) that would encourage some (or all) apps to upgrade immediately. Similarly, the use of survival models helps us measure *"by what %"* individual covariates will predict a delay in upgrade but does not measure *"by how much time"*. While the Kaplan-Meier analysis suggests that overreaching apps delay upgrading by about 3 months on average, future research could incorporate RCTs to measure the delay time (measured in months) that new privacy laws can cause.

5.3 Contributions to theory and practice

Limitations notwithstanding, our study makes a number of significant contributions. Our study adds to a small but growing stream of literature that analyze the consequences of platform policy changes. Given the multi-sided nature of the platform, any policy changes targeting one side of the platform (e.g., app users)

will have unintended consequences on the other side (e.g., app developers). In our context, Android designed a new mechanism to improve the privacy choices for consumers. However, this decision had consequences on app developers that decided to strategically delay upgrading. On a related note, this research adds to the literature on informed consent by quantifying the unintended consequence of enforcing tighter consent policies on sellers. When newer policies are introduced or existing policies are changed to mitigate users' privacy concerns, it can aggravate developers' decision-making burden and may lead to sub-optimal decisions (e.g., in our context, deciding to delay upgrade). We also add to the literature on platform strategy by suggesting that a fragmented platform such as Android should carefully design the upgrade window to minimize sub-optimal outcomes for the participants (apps, users) or the platform.

Next, this study also informs literature on digital platforms which investigates provider's responses when new privacy policies are imposed on them. Literature has documented sellers' response to changes in the nature of market competition (Chen and Forman 2006, Foerderer et al. 2018), however these responses are voluntary and not in reaction to policy change. Given the nature of policy adoption, quantifying the antecedents and outcomes of delaying policy adoption is very useful for both policy makers and the providers. By showing that even popular apps are negatively impacted if they strategically continue overreaching, we highlight the role of platform mechanisms under a power asymmetry in safeguarding users' interest.

Finally, this research makes a methodological contribution to the literature on privacy preferences in the sub domain of information economics. By combining advanced text-mining techniques with statistical techniques such as Peer Group Analysis or Permission Prevalence Analysis, we draw from a rich stream of literature on computer security and privacy. Our use of modern text analysis techniques such as Word2Vec or topic modeling techniques help uncover the mechanisms from rich, unstructured text data, which help guide our investigation.

References

Acquisti A, Grossklags J. (2003) Losses, Gains, and Hyperbolic Discounting: An Experimental Approach to Information Security Attitudes and Behavior. *Proc. 2nd Annual Workshop on Econom. and Inform. Security*. 1-27

Acquisti A, Grossklags J (2005) Privacy and Rationality in Individual Decision Making. *IEEE Security & Privacy*. 3(1): 26-33.

Adamopoulos P, Ghose A, Todri V (2018) The impact of user personality traits on word of mouth: Text-mining social media platforms. *Information Systems Research*, *29*(3), pp.612-640.

Adjerid I, Acquisti A, Telang R, Padman R, Adler-Milstein J (2016) The impact of privacy regulation and technology incentives: The case of health information exchanges. *Management Sci.*, *62*(4), pp.1042-1063.

Allison P (2002) Bias in fixed-effects Cox regression with dummy variables. *Manuscript, Department of Sociology, University of Pennsylvania*.

Almuhimedi H, Schaub F, Sadeh N, Adjerid I, Acquisti A, Gluck J, Cranor LF, Agarwal Y (2015) Your location has been shared 5,398 times! A field study on mobile app privacy nudging. *Proc. 33rd annual ACM Conf. on human factors in computing Sys.* (pp. 787-796).

Al-Hussaini F (2020) Mobile Ads: Can Your Phone Hear Your Conversations? *Retrieved on 8th March 2021* from https://www.spiralytics.com/blog/mobile-ads-can-phone-hear-conversations-infographic/

Android. (2016) Distribution Dashboard. Retrieved on May 1, 2016 form https://developer.android.com/about/ dashboards/.

Android. (2021) Meet Google Play's target API level requirement retrieved on October 21st 2021 form https://developer.android.com/distribute/best-practices/develop/target-sdk

Angst CM, Wowak KD, Handley SM, Kelley K. (2017) Antecedents of information systems sourcing strategies in US hospitals: A longitudinal study. *MIS Quarterly*, *41*(4): 1129-1152.

Aridor G, Che YK, Salz T. (2020) The economic consequences of data privacy regulation: Empirical evidence from GDPR. *NBER working paper*, (w26900).

Askalidis, G., 2015. The impact of large scale promotions on the sales and ratings of mobile apps: Evidence from Apple's App Store. arXiv preprint arXiv:1506.06857.

Austin PC, Small DS (2014) The use of bootstrapping when using propensity-score matching without replacement: a simulation study. *Statist. in Medicine*. 33(24): 4306-4319.

Austin, PC, (2017) A tutorial on multilevel survival analysis: methods, models and applications. *International Statistical Review*, *85*(2):185-203.

Autor DH (2003) Outsourcing at Will: The Contribution of Unjust Dismissal Doctrine to the Growth of Employment Outsourcing. *J. Labor Econom.* 21(1): 1-42.

Bankhead P (2018) Providing a Safe and Secure Experience for our Users. *Retrieved on 8th March 2021* from https://android-developers.googleblog.com/2018/10/providing-safe-and-secure-experience.html

Bapna R, Umyarov A (2015) Do Your Online Friends Make You Pay? A Randomized Field Experiment in an Online Music Social Network. Management Sci. 61(8): 1902-1920.

Bapna R, Ramaprasad J, Shmueli G, Umyarov A (2016) One-way mirrors in online dating: A randomized field experiment. Management Science, 62(11), pp.3100-3122

Barkhuus L, Dey A. (2003) Location-Based Services for Mobile Telephony: a study of users' privacy concerns. Interact. 3: 702-212.

Bayus BL (1997) Speed-to-market and new product performance trade-offs. J. Product Innovation Management, 14(6), pp.485-497.

Bertrand M, Duflo E, Mullainathan S (2004) How much should we trust Differences-in-Differences estimates? Quart J. Econom. 119(1): 249-275.

Bertschek I, Kesler R. (2017) Let the user speak: Is feedback on Facebook a source of firms' innovation?. *ZEW-Centre for European Economic Research Discussion Paper*, (17-015).

Brown C, Dinc IS (2005) The politics of bank failures: Evidence from emerging markets. The Quarterly Journal of Economics, 120(4), pp.1413-1444.

Brandimarte L, Acquisti A, Loewenstein G. (2013) Misplaced confidences: Privacy and the control paradox. *Social Psychological and Personality Science*, *4*(3), pp.340-347.

Bresnahan T, Orsini J, Yin PL. (2014) Platform choice by mobile app developers. *NBER working paper*.

Cadwalladr C, Graham-Harrison E. (2018) Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. *The Guardian. Retrieved on 8th March 2021* from https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election.

Campbell J, Goldfarb A, Tucker CE (2015) Privacy regulation and market structure. *J. Econom. & Management Strategy*, *24*(1), pp.47-73.

Chen P-y, Forman C (2006) Can vendors influence switching costs and compatibility in an environment with open standards? *MIS Quart.* 30: 541-562.

Cimpanu C. (2018) Google restricts which Android apps can request Call Log and SMS permissions. *ZDNet* (October 9),https://www.zdnet.com/article/google-restricts-which-android-apps-can-request-call-log-and-sms-permissions/.

Comino S, Manenti FM, Mariuzzo F. (2016) To upgrade or not to upgrade? the release of new versions to survive in the hypercompetitive app market. In *Proceedings of the International Workshop on App Market Analytics*: 37-42.

Dehejia RH, Wahba S (2002) Propensity Score-Matching methods for nonexperimental causal studies. *Review of Economics and statistics*. 84(1): 151-161.

Erlingsson U (2016) Data-driven software security: Models and methods. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)* (pp. 9-15). IEEE.

Ershov D. (2018) The effect of consumer search costs on entry and quality in the mobile app market. Unpublished manuscript.

Fang VW, Tian X, Tice S (2014) Does Stock Liquidity Enhance or Impede Firm Innovation?. *J. Finance*. 69(5): 2085-2125.

Farahat A, Bailey MC (2012) How effective is targeted advertising?. *Proc. 21st Internat. Conf. World Wide Web* (pp. 111-120).

Felt AP, Chin E, Hanna S, Song D, Wagner D. (2011). Android Permissions Demystified. *Proc.18th ACM Conf. Comput. and Comm.* 627-638.

Fishman A, Rob R (2000) Product innovation by a durable-good monopoly. *RAND J. Econom.*, pp.237-252.

Fleder D, Hosanagar K. (2009) Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. Management Sci, (*55:5*), pp.697-712.

Fudenberg D, Tirole J (1998) Upgrades, tradeins, and buybacks. *RAND J. Econom.*, pp.235-258.

Foerderer J, Kude T, Mithas S, Heinzl A (2018) Does Platform Owner's Entry Crowd Out Innovation? Evidence From Google Photos. *Inform. Systems Res.* 29(2): 444-460.

Gao H, Zhang W (2016) Employment Nondiscrimination Acts and Corporate Innovation. *Management Sci.* 63(9): 2982-2999.

Garber J (2018) GDPR–compliance nightmare or business opportunity? *Comput. Fraud & Security*. 2018(6): 14-15.

Garg R, Telang R (2013) Inferring app demand from publicly available data. *MIS Quart.* 37(4): 1253-1264.

Ghose A, Han SP (2014) Estimating Demand for Mobile Applications in the New Economy. *Management Sci.* 60(6): 1470-1488.

Gkatzelis V, Aperjis C, Huberman BA. (2015) Pricing private data. *Electronic Markets*, 25(2), pp.109-123.

Goldfarb A, Tucker CE (2011) Privacy regulation and online advertising. *Management Sci.*, *57*(1), pp.57-71.

Goldfarb A, Tucker C. (2012) Privacy and innovation. *Innovation Policy and the Economy*, 12(1), pp.65-90.

Golubev P (2019) How advertisers learn which Android apps you use. *Retrieved on 8th March 2021* from https://usa.kaspersky.com/blog/android-device-identifiers/20040/

Goode L. (2018) App Permissions Don't Tell Us Nearly Enough About Our Apps. *Wired* (August 14), https://www.wired.com/story/app-permissions/.

Grace MC, Zhou W, Jiang X, Sadeghi A-R. (2012) Unsafe Exposure Analysis of Mobile In-App Advertisements. *Proc.5th ACM Conf. Security and Privacy in Wireless and Mobile Networks*. 101-112.

Gradwohl R. (2018) Firms quick to adopt EU data regs will have first-mover advantage. *The Hill* (May 25), https://thehill.com/opinion/cybersecurity/389405-firms-quick-to-adopt-eu-data-regs-will-have-first-mover-advantage

Hann I-H, Hui K-L, Lee S-YT, Png IP (2007) Overcoming Online Information Privacy Concerns: An Information-Processing Theory Approach. *J. Management Inform. Systems*. 24(2): 13-42.

Heddings L (2018) Facebook is Using Your Phone Number to Target Ads and You Can't Stop It. *Retrieved on 8th March 2021* from https://www.howtogeek.com/367766/facebook-is-using-your-phone-number-to-target-ads-and-you-can%e2%80%99t-stop-it/

Hinneburg A, Keim DA. (1999) Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering. *Proc.25th Internat. Conf. Very Large Databases*. 506-517.

Hoofnagle CJ, King J, Li S, Turow J (2010) How Different are Young Adults from Older Adults When it Comes to Information Privacy Attitudes and Policies? SSRN Working Paper.

IAB. (2021) Content Taxonomy Retrieved on 21st October 2021 from https://iabtechlab.com/standards/content-taxonomy/

Jabr W, Gutt D, Neumann J, Kundisch D (2020) The Fate of the App: Economic Implications of Updating under Reputation Resetting. *Available at SSRN 3565937*.

Jana S, Erlingsson Ú, Ion I. (2015) Apples and oranges: Detecting least-privilege violators with peer group analysis. *arXiv preprint arXiv:1510.07308*.

Jain, DC, Vilcassim, NJ (1991) Investigating household purchase timing decisions: A conditional hazard function approach. *Marketing Sci.*, *10*(1), pp.1-23.

Jedrzejczyk L, Price BA, Bandara AK, Nuseibeh B (2010) On the impact of real-time feedback on users' behaviour in mobile location-sharing applications. *Proc. Sixth Symposium on Usable Privacy and Security* (pp. 1-12).

Jensen C, Potts C. (2004) Privacy policies as decision-making tools: An evaluation of online privacy notices. *Proc. SIGCHI Conf. Human Factors in Comput. Systems*. 471-478.

Jia J, Jin GZ, Wagman L. (2021) The Short-Run Effects of the General Data Protection Regulation on Technology Venture Investment. *Marketing Science*.

Johnson PM, Kou H, Paulding M, Zhang Q, Kagawa A, Yamashita T. (2005) Improving software development management through software project telemetry. IEEE software, 22(4), pp.76-85.

Johnson GA, Shriver SK, Du S. (2020) Consumer privacy choice in online advertising: Who opts out and at what cost to industry?. *Marketing Sci.*, 39(1), pp.33-51.

Johnson GA, Shriver SK, Goldberg S. (2021) Privacy & market concentration: Intended & unintended consequences of the GDPR. *Available at SSRN 3477686*.

Kane GC, Ransbotham S (2016) Research note—Content and Collaboration: An Affiliation Network Approach to Information Quality in Online Peer Production Communities. *Inform. Systems Res.* 27(2), 424-439.

Kauffman RJ, McAndrews J, Wang Y-M (2000) Opening the "black box" of network externalities in network adoption. *Inform. Systems Res.* 11(1): 61-82.

Kechagia M, Mitropoulos D, Spinellis D. (2015) Charting the API minefield using software telemetry data. *Empirical Software Engineering* 20(6),1785-1830.

Kelley PG, Consolvo S, Cranor LF, Jung J, Sadeh N, Wetherall D. (2012) A Conundrum of Permissions: Installing Applications on an Android Smartphone. *Internat. Conf. Financial Cryptography and Data Security*. 68-79

Kesler R, Kummer ME, Schulte P. (2017) Mobile applications and access to private data: The supply side of the Android ecosystem. *ZEW-Centre for European Economic Research Discussion Paper*, (17-075).

Kesler R, Kummer ME, Schulte P. (2019) Competition and Privacy in Online Markets: Evidence from the Mobile App Industry.

Kröger JL, Raschke P (2019) Is my phone listening in? On the feasibility and detectability of mobile eavesdropping. In *IFIP Annual Conf. Data and Applications Security and Privacy* (pp. 102-120). Springer, Cham.

Kumar, A., Smith, M.D., Telang, R. (2014) Information discovery and the long tail of motion picture content. MIS Quarterly, 38(4), pp.1057-1078.

Kummer ME, Schulte P. (2019) When private information settles the bill: Money and privacy in Google's market for smartphone applications. *Management Sci.*, 65(8), 3470-3494.

Lambrecht A. (2017) E-Privacy provisions and venture capital investments in the EU.

Lambrecht A, Tucker CE. (2013) When does retargeting work? Information specificity in online advertising. *J. Marketing Res.*, *50*(5), pp.561-576.

Leyden BT (2019) There's an app (update) for that: Understanding product updating under digitization. *Working paper, Cornell University*.

Leyden BT (2021) Platform Design and Innovation Incentives: Evidence from the Product Ratings System on Apple's App Store.

Li X, Bresnahan T, Yin PL. (2016) Paying incumbents and customers to enter an industry: Buying downloads. *SSRN Working Paper 2834564*.

Light A. (1995) Hazard model estimates of the decision to reenroll in school. *Labour Economics*, *2*(4), pp.381-406.

Limer E. (2018) Hundreds of Apps Can Eavesdrop Through Phone Microphones to Target Ads. *Popular Mechanics.* Retrieved on *8th March 2021* from https://www.popularmechanics.com/technology/security/a14533262/ alphonso-audio-ad-targeting/.

Liu B, Andersen MS, Schaub F, Almuhimedi H, Zhang SA, Sadeh N, Agarwal Y and Acquisti A .2016. Follow my recommendations: A personalized privacy assistant for mobile app permissions. In *Twelfth Symposium on Usable Privacy and Security ({SOUPS} 2016)* (pp. 27-41).

Liu X, Lynch L (2011) Do agricultural land preservation programs reduce farmland loss? Evidence from a propensity score matching estimator. *Land Economics*. 87(2): 183-201.

Mahajan V, Muller E (1996) Timing, diffusion, and substitution of successive generations of technological innovations: The IBM mainframe case. *Technological Forecasting and Social Change*, *51*(2), pp.109-132.

Mahajan V, Muller E (1998) When Is It Worthwhile Targeting the Majority Instead of the Innovators in a New Product Launch? *Journal of Marketing Research*. 35(4): 488-495.

Maheshwari S. (2017) That Game on Your Phone May Be Tracking What You're Watching on TV. *The New York Times* Retrieved on *8th March 2021* from https://www.nytimes.com/2017/12/28/business/media/alphonso-app-tracking.html

Mayya R, Ye S, Viswanathan S, Agarwal R (2021) Who Forgoes Screening in Online Markets and When? Evidence from Airbnb. *MIS Quart.* 45(4):1745-1776.

McIlroy S, Ali N, Hassan AE (2016) Fresh apps: an empirical study of frequently-updated mobile apps in the Google play store. *Empirical Software Engineering*, *21*(3), pp.1346-1370.

Mehra A, Seidmann A, Mojumder P (2014) Product life-cycle management of packaged software. *Production and Operations Management*, *23*(3), pp.366-378.

Meyer BD (1995) Natural and Quasi-experiments in Economics. *J. Bus. & Econom. Statist.* 13(2): 151-161.

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Inform. Processing Sys.* 3111-3119.

Miller AR, Tucker CE (2009) Privacy protection and technology diffusion: The case of electronic medical records. *Management Sci.*, *55*(7), pp.1077-1093.

Miller AR, Tucker CE (2018) Privacy protection, personalized medicine, and genetic testing. *Management Sci.*, *64*(10), pp.4648-4668.

Mnih A, Hinton GE. (2009). A scalable hierarchical distributed language model. *Advances in neural information processing Sys.*. 1081-1088.
Montgomery AL, Li S, Srinivasan K, Liechty JC. (2004) Modeling online browsing and path analysis using clickstream data. *Marketing Sci.*, 23(4), pp.579-595.
Oestreicher-Singer G, Sundararajan A (2012) Reccomendation Networks and the Long Tail of Electronic Commerce. *MIS Quart*. 36(1):65-83.
Page L, Brin S, Motwani R, Winograd T (1999) The PageRank citation ranking: Bringing order to the web. *Stanford InfoLab*.
Pelikan M, Hogben G, Erlingsson U (2017) Identifying Intrusive Mobile Apps Using Peer Group Analysis. *Google Security Blog* (July 12), https://security.googleblog.com/2017/07/identifying-intrusive-mobile-apps-using.html
Peddinti, S.T., Bilogrevic, I., Taft, N., Pelikan, M., Erlingsson, Ú., Anthonysamy, P. and Hogben, G (2019) Reducing permission requests in mobile apps. In *Proc. Internet Measurement Conf.* (pp. 259-266).
Peng H, Gates C, Sarma B, Li N, Qi Y, Potharaju R, Nita-Rotaru C, Molloy I .2012. Using probabilistic generative models for ranking risks of android apps. *Proc. 2012 ACM Conf. on Computer and Communications Security* (pp. 241-252).
Peukert, C., Bechtold, S., Batikas, M. and Kretschmer, T., 2020. European privacy law and global markets for data. *Available at SSRN 3560392*.
Pingitore G, Rao V, Cavallaro K, Dwivedi K. (2017) To Share or Not to Share. *Deloitte University Press*.
Rainie L, Kiesler S, Kang R, Madden M, Duggan M, Brown S, Dabbish L (2013) Anonymity, Privacy, and Security Online. *Pew Res. Center*. 5.
Resnick P, Zeckhauser R (2002) Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system. *Econom. Internet and e-commerce*. 127-157.
Romanosky S, Telang R, Acquisti A (2011) Do data breach disclosure laws reduce identity theft?. *J. Policy Analysis and Management*, *30*(2), pp.256-286.
Sarma BP, Li N, Gates C, Potharaju R, Nita-Rotaru C, Molloy I. (2012) Android Permissions: A Perspective Combining Risks and Benefits. *Proc. 17th ACM Sympos. Access Control Models and Tech*. 13-22
Savage, SJ, Waldman DM (2015) Privacy tradeoffs in smartphone applications. *Economics Letters* (137), pp.171-175.
Seetharaman, P.B. and Chintagunta, P.K., (2003) The proportional hazard model for purchase timing: A comparison of alternative specifications. *J. Business & Econom. Statistics*, *21*(3), pp.368-382.
Smith JA, Todd PE (2005) Does Matching Overcome Lalonde's Critique of Nonexperimental Estimators? *J. Econometrics*. 125(1-2): 305-353.
Spiekermann S, Grossklags J, Berendt B. (2001) E-privacy in 2nd Generation e-commerce: Privacy preferences versus actual Behavior. *Proc. 3rd ACM Conf. Electronic Commerce*. 38-47
Stamm SM, Tripp; Kuronen, Jessica. (2018) How Pizza Night Can Cost More in Data Than Dollars. *The Wall Street Journal*. Retrieved on *8th March 2021* from https://www.wsj.com/graphics/how-pizza-night-can-cost-more-in-data-than-dollars/
Statista. (2021a) Android version market share distribution among smartphone owners. Retrieved on *8th March 2021* from https://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/.
Statista. (2021b). Android operating system share worldwide by OS version from 2013 to 2020. Retrieved on *8th March 2021* from https://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/
Tambe P, Hitt LM, Brynjolfsson E (2012) The extroverted firm: How external information practices affect innovation and productivity. *Management Sci.*, 58(5), pp.843-859.
Taylor VF, and Martinovic I (2016) Securank: Starving permission-hungry apps using contextual permission analysis. *Proc. 6th Workshop on Security and Privacy in Smartphones and Mobile Devices*, 43-52.
Taylor VF, and Martinovic I (2017) To update or not to update: Insights from a two-year study of android app evolution. *Proc. 2017 ACM on Asia Conf. on Computer and Communications Security* (pp. 45-57).

Tiffany K (2018) Advertisers can easily track you, your kid, your doctor, and the president. *Retrieved on 8th March 2021* from https://www.vox.com/the-goods/2018/12/11/18136361/location-tracking-data-ad-targeting-facebook-google-amazon

Towe, C. and Lawley, C., (2013) The contagion effect of neighboring foreclosures. *American Econom. Journal: Econom. Policy*, *5*(2), pp.313-35.

Watanabe T, Akiyama M, Kanei F, Shioji E, Takata Y, Sun B, Ishi Y, Shibahara T, Yagi T, Mori T. (2017) Understanding the origins of mobile app vulnerabilities: A large-scale measurement study of free and paid apps. *Proc. 14th Internat. Conf. Mining Software Repositories*. 14-24.

Weeks C (2018) Guide to Android App Permissions & How to Use Them Smartly. *Retrieved on 8th March 2021* from https://www.avg.com/en/signal/guide-to-android-app-permissions-how-to-use-them-smartly

Wei X, Gomez L, Neamtiu I, Faloutsos M. (2012) Permission evolution in the android ecosystem. In *Proc. 28th Annual Computer Security Applications Conference* (pp. 31-40).

Wijesekera P, Baokar A, Hosseini A, Egelman S, Wagner D, Beznosov K (2015) Android permissions remystified: A field study on contextual integrity. *24th {USENIX} Security Symposium ({USENIX} Security 15)* (pp. 499-514).

Wilson LO, Norton JA (1989) Optimal entry timing for a product line extension. *Marketing Sci*. *8*(1), pp.1-17

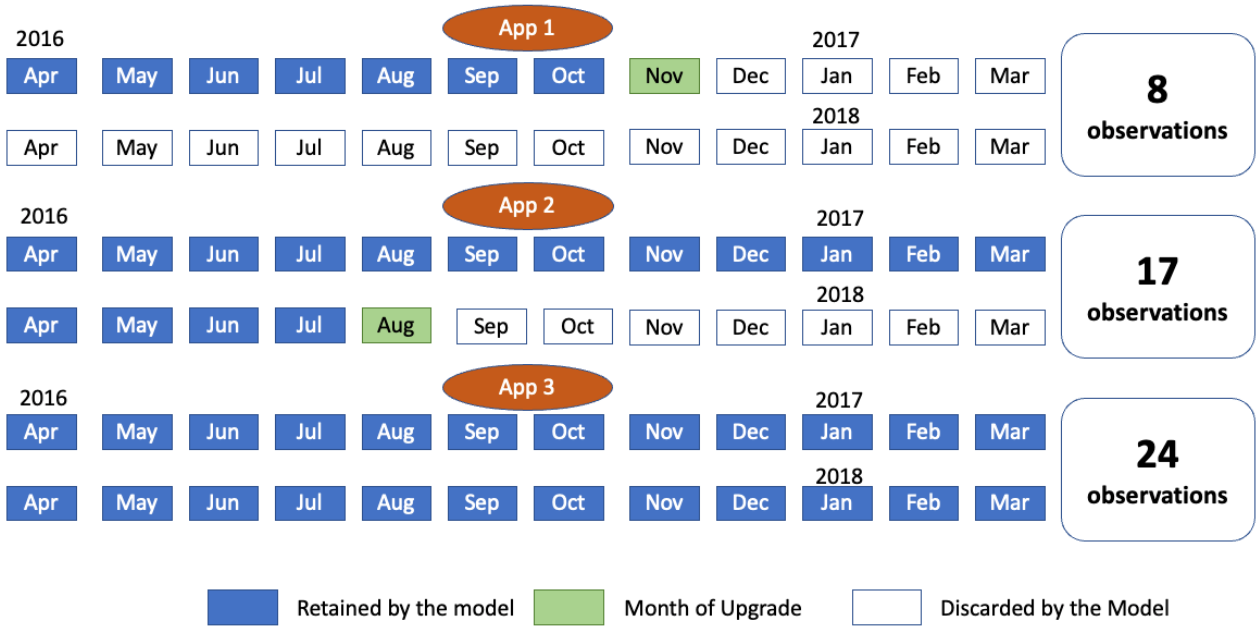
Xing L, Pan X, Wang R, Yuan K, Wang X. (2014). Upgrading your android, elevating my malware: Privilege escalation through mobile os updating. In *2014 IEEE symposium on security and privacy*: 393-408.

Yin PL, Davis JP, Muzyrya Y (2014) Entrepreneurial innovation: Killer apps in the iPhone ecosystem. *American Econom. Review*, 104(5), pp.255-59.

Zimmer JC, Arsal RE, Al-Marzouq M, Grover V (2010) Investigating online information disclosure: Effects of information relevance, trust and risk. *Inform. & Management*. 47(2): 115-123.

Appendix A

Appendix Figure A1. Survival model sample demonstration



Note: In this example, only 49 observations (8+17+24) are retained in the model out of a total of 72 observations.

Table A1. Android 6.0 Changes

Panel A: List of Runtime Permissions according to Android Platform

Sl. No	Permission Name	Permission Group
1	READ_CALENDAR	CALENDAR
2	WRITE_CALENDAR	CALENDAR
3	READ_CALL_LOG	CALL_LOG
4	WRITE_CALL_LOG	CALL_LOG
5	PROCESS_OUTGOING_CALLS	CALL_LOG
6	CAMERA	CAMERA
7	READ_CONTACTS	CONTACTS
8	WRITE_CONTACTS	CONTACTS
9	GET_ACCOUNTS	CONTACTS
10	ACCESS_FINE_LOCATION	LOCATION
11	ACCESS_COARSE_LOCATION	LOCATION
12	RECORD_AUDIO	MICROPHONE
13	READ_PHONE_STATE	PHONE
14	READ_PHONE_NUMBERS	PHONE
15	CALL_PHONE	PHONE
16	ANSWER_PHONE_CALLS	PHONE
17	ADD_VOICEMAIL	PHONE
18	USE_SIP	PHONE
19	BODY_SENSORS	SENSORS
20	SEND_SMS	SMS
21	RECEIVE_SMS	SMS
22	READ_SMS	SMS
23	RECEIVE_WAP_PUSH	SMS
24	RECEIVE_MMS	SMS
25	READ_EXTERNAL_STORAGE	STORAGE
26	WRITE_EXTERNAL_STORAGE	STORAGE

Panel B: Changes made to existing Android Features in Android 6.0

Change Type	Brief Description of change
Runtime Permissions	Changed how apps seek permission to access users' information
Doze and App Standby	Power saving optimization for idle devices and apps
Apache HTTP Client Removal	Apps can instead use an efficient class, *HttpURLConnection*
BoringSSL	Android moved from OpenSSL to BoringSSL library
Access to Hardware Identifier	Apps can use *access fine location* permission instead
Notifications	Improvements to how notifications can be updated
AudioManager	Use alternative methods to setting volumes and muting
Text Selection	Makes cut, copy, paste easier
Browser Bookmark	Apps are required to store bookmarks data internally (instead of global)
Android Keystore	Security enhancement of keystore provider
Wi-Fi and Networking	Cannot alter Wi-Fi Configuration created by other apps. Provide new APIs for network binding
Camera Service	Improvement to Camera API
USB Connection	Default connections through USB are in charge-only mode
Android for Work	Changes in Android for Work (device policies etc.)
APK Validation	Stricter validation of APK files to ensure there are no corruption

List retrieved from https://developer.android.com/about/versions/marshmallow/android-6.0-changes

Panel C: New Features introduced in Android 6.0

New Feature	Brief Description of the New feature
Fingerprint Authentication	Allows apps to use fingerprints on supported phones
Confirm Credential	Authenticate users based on how recently they last unlocked phone
App Linking	Allows apps to associate with a web domain they own
Auto Backup for Apps	Allows apps for automatic full data backup
Direct Share	Allow apps to use APIs to share content with Android users
Voice Interactions	Improves conversational voice experience
Assist API	Allows apps to engage with users through an assistant
Adoptable Storage Devices	Users can adopt external storage devices
Notifications	Improves notification
Bluetooth Stylus Support	Added support for Bluetooth Stylus
Bluetooth low energy scanning	Improves power efficiency
Hotspot Improvement	Allows Hotspot 2.0
4K Display Mode	Allows compatible hardware to have 4K display resolution
Themeable Colors	Supports Theme attributes
Audio Features	A few improvements on Audio API
Video Features	A few improvements in Video API
Camera Features	A few new features such as Flashlight, reprocessing
Android for Work Features	A few new APIs for Android for Work

List retrieved from https://developer.android.com/about/versions/marshmallow/android-6.0

Table A2. App Categories

Category Group	Group Description	Categories in the Group
Content Consumption	Apps that allow consumption of content on a mobile platform	Comics Entertainment Media and Video Music and Audio Sports Video Players
Learn and Explore	Apps that allow users to learn and explore content on the mobile platform	Art and Design Books and References Education Medical News and Magazines Travel and Local Weather Parenting Libraries and Demo
Personal	Apps that are personal to users	Beauty Dating Health and Fitness Lifestyle Personalization Social
Utility – Mobile Access	These are the utility apps that have a major offline/web presence and provides online access through the platform	Business Auto and Vehicles Events Finance House and Home Shopping Transportation
Utility – Mobile Specific	These are utility apps that are present mainly on the mobile platform. Its existence depends on features of smartphone	Maps and Navigation Food and Drink Communication Tools Photography Productivity
Games	These are apps that are categorized as games by Google Play Store	Action Game Adventure Game Arcade Game Board Game Casino Game Casual Game Educational Game Music Game Puzzle Game Racing Game Role Playing Game Simulation Game Sports Game Strategy Game Trivia Game Word Game

Table A3. Analysis of Upgrading to Latest Version of Android

Dependent variable - $upgrade_{it}$	Model (1)		Model (2)		Model (3)		Model (4)	
Model Variant	*Cox PH Model*		*Mixed Effects Survival Model*		*Cox PH Model*		*Mixed Effects Survival Model*	
	Hazard Ratio		Hazard Ratio		Hazard Ratio		Hazard Ratio	
$runtime_permissions_{it}$	0.994***	(0.001)	0.992***	(0.001)				
$runtime_permissions_count_{it}$					0.953***	(0.007)	0.924***	(0.010)
$\Delta rating_count_{it}$	1.000	(0.000)	1.000***	(0.000)	1.000	(0.000)	1.000***	(0.000)
$rating_{it}$	0.981	(0.031)	0.988	(0.047)	1.036	(0.033)	1.041	(0.050)
$log(popularity_{it})$	0.978*	(0.011)	1.221***	(0.018)	0.965**	(0.010)	1.211***	(0.018)
$games_i$	0.930	(0.038)	1.009	(0.060)	0.836***	(0.034)	0.865*	(0.053)
$personal_apps_i$	0.892*	(0.041)	1.007	(0.067)	0.917	(0.042)	1.061	(0.073)
$utility_mobilespecific_i$	0.826***	(0.037)	0.755***	(0.051)	0.886**	(0.039)	0.838**	(0.057)
$utility_mobileaccess_i$	0.763***	(0.040)	0.811**	(0.065)	0.844**	(0.045)	0.963	(0.080)
$learn_explore_i$	0.966	(0.044)	1.022	(0.069)	0.951	(0.043)	1.011	(0.070)
$category_i$= $content_consumption_i$ (baseline)								
$dayssinceupdate_{it}$	0.954***	(0.001)	0.955***	(0.001)	0.954***	(0.001)	0.955***	(0.001)
$screenshots_{it}$	1.003	(0.002)	1.022***	(0.004)	1.001	(0.002)	1.019***	(0.004)
$log(filesize_{it})$	1.072***	(0.017)	1.166***	(0.027)	1.127***	(0.018)	1.250***	(0.030)
$developer_appcount_{it}$	1.000	(0.000)	0.998***	(0.000)	1.000*	(0.000)	0.998***	(0.000)
$installtime_permissions_count_{it}$					0.971***	(0.004)	0.963***	(0.006)
$ln(p)$			161.709***	(1.494)			162.399***	(1.494)
$var(packageid)$			163.946***	(27.673)			156.979***	(26.379)
Count of Apps	11,675		11,675		11,675		11,675	
Download Bucket Dummies	YES		YES		YES		YES	
Log Likelihood	-56301.34		-25460.72		-56173.05		-25388.91	
AIC	112636.7		50961.44		112382.1		50819.81	
BIC	112806		51160.7		112561.4		51029.04	
# of Obs.	156,903		156,903		156,903		156,903	
X^2	2131.82		1863.70		2505.43		2211.98	
p	0.000		0.000		0.000		0.000	

*** p<0.001, ** p<0.01, * p<0.05; Clustered standard errors in parentheses.

Table A4. Analysis of Upgrading to Latest Version of Android –*essential/nonessential* permissions based on IAB categories

Dependent variable - *upgrade*$_{it}$	Model (1)		Model (2)		Model (3)		Model (4)		Model (5)		Model (6)	
Model Variant	*Cox PH Model*		*Mixed Effects Survival Model*		*Cox PH Model*		*Mixed Effects Survival Model*		*Cox PH Model*		*Mixed Effects Survival Model*	
	Hazard Ratio		Hazard Ratio		Hazard Ratio		Hazard Ratio		Hazard Ratio		Hazard Ratio	
essential_ permissions$_{it}$	0.999	(0.002)	1.013***	(0.003)								
nonessential_ permissions$_{it}$	0.993***	(0.001)	0.990***	(0.001)								
essential_ permissions_count$_{it}$					0.969	(0.020)	1.138***	(0.035)				
nonessential_ permissions_count$_{it}$					0.952***	(0.007)	0.908***	(0.010)				
overreached$_i$ *(1/0)*									0.762***	(0.021)	0.604***	(0.024)
Δrating_count$_{it}$	1.000	(0.000)	1.000***	(0.000)	1.000	(0.000)	1.000***	(0.000)	1.000	(0.000)	1.000***	(0.000)
rating$_{it}$	0.981	(0.031)	0.993	(0.047)	1.036	(0.033)	1.046	(0.050)	1.003	(0.032)	1.019	(0.048)
log(popularity$_{it}$*)*	0.978*	(0.011)	1.219***	(0.018)	0.965**	(0.010)	1.209***	(0.018)	0.977*	(0.011)	1.219***	(0.018)
games$_i$	0.957	(0.040)	1.119	(0.069)	0.844***	(0.036)	0.971	(0.062)	0.910*	(0.037)	0.957	(0.058)
personal_apps$_i$	0.902*	(0.041)	1.047	(0.070)	0.920	(0.042)	1.092	(0.074)	0.881**	(0.040)	0.991	(0.067)
utility_mobilespecific$_i$	0.834***	(0.037)	0.770***	(0.052)	0.889**	(0.039)	0.858*	(0.058)	0.822***	(0.037)	0.763***	(0.051)
utility_mobileaccess$_i$	0.761***	(0.040)	0.801**	(0.064)	0.841**	(0.044)	0.917	(0.076)	0.754***	(0.039)	0.815*	(0.065)
learn_explore$_i$	0.974	(0.044)	1.052	(0.071)	0.953	(0.043)	1.033	(0.072)	0.953	(0.043)	0.997	(0.068)
category$_i$*= content_consumption*$_i$ *(baseline)*												
dayssinceupdate$_{it}$	0.954***	(0.001)	0.955***	(0.001)	0.954***	(0.001)	0.955***	(0.001)	0.954***	(0.001)	0.955***	(0.001)
screenshots$_{it}$	1.003	(0.002)	1.021***	(0.004)	1.001	(0.002)	1.018***	(0.004)	1.003	(0.002)	1.022***	(0.004)
log(filesize$_{it}$*)*	1.070***	(0.017)	1.157***	(0.027)	1.125***	(0.018)	1.231***	(0.029)	1.079***	(0.017)	1.193***	(0.028)
developer_appcount$_{it}$	1.000	(0.000)	0.998***	(0.000)	1.000*	(0.000)	0.998***	(0.000)	1.000	(0.000)	0.998***	(0.000)
installtime_permissions_count$_{it}$					0.971***	(0.004)	0.964***	(0.006)				
ln(p)			172.239***	(1.585)			170.518***	(1.635)			172.160***	(1.595)
var(packageid)			177.266***	(31.185)			135.049***	(23.372)			170.283***	(29.942)
Count of Apps	11,675		11,675		11,675		11,675		11,675		11,675	
Download Bucket Dummy	YES		YES		YES		YES		YES		YES	
Log Likelihood	-56298.36		-25450.33		-56172.73		-25376.97		-56283.11		-25440.02	
AIC	112632.7		50942.67		112383.5		50797.95		112600.2		50920.05	
BIC	112812.1		51151.9		112572.8		51017.14		112769.6		51119.32	
# of Obs.	156,903		156,903		156,903		156,903		156,903		156,903	
X^2	2141.57		1947.58		2505.26		2276.55		2237.64		2045.44	
p	0.000		0.000		0.000		0.000		0.000		0.000	

*** p<0.001, ** p<0.01, * p<0.05; Clustered standard errors in parentheses.

Essential and nonessential permission ratio are the key independent variables in columns (1) and (2)

Count of essential and nonessential permissions are the key independent variables in columns (3) and (4)

A dichotomous variable representing whether an app *overreached* is the key independent variables in columns (5) and (6)

Table A5. Analysis of Upgrading to Latest Version of Android

Dependent variable - $upgrade_{it}$	(1)		(2)	
	Point Estimates		*Hazard Ratio*	
$location_coarse_{it}$	0.106**	(0.035)	1.112**	(0.039)
$location_precise_{it}$	-0.146***	(0.038)	0.864***	(0.033)
$access_bodysensors_{it}$	0.361	(0.349)	1.435	(0.501)
$make_phonecall_{it}$	-0.044	(0.061)	0.957	(0.058)
$access_camera_{it}$	-0.012	(0.037)	0.989	(0.037)
$find_accounts_{it}$	-0.001	(0.027)	0.999	(0.027)
$reroute_outgoingcalls_{it}$	0.005	(0.132)	1.005	(0.132)
$read_phonestatus_and_identity_{it}$	-0.392***	(0.026)	0.675***	(0.018)
$read_sms_{it}$	-0.035	(0.111)	0.966	(0.107)
$receive_mms_{it}$	0.012	(0.089)	1.012	(0.090)
$record_audio_{it}$	-0.170***	(0.048)	0.844***	(0.040)
$send_sms_{it}$	-0.214*	(0.105)	0.807*	(0.085)
$make_sip_call_{it}$	0.460	(0.319)	1.584	(0.505)
$add_voicemail_{it}$	-45.136	(0.000)	0.000	(0.000)
$read_write_phonememory_{it}$	-0.082**	(0.029)	0.921**	(0.027)
$read_write_calendar_{it}$	-0.010	(0.079)	0.990	(0.078)
$read_write_calllog_{it}$	-0.556***	(0.110)	0.574***	(0.063)
$read_write_contact_{it}$	0.067	(0.049)	1.069	(0.053)
$\Delta rating_count_{it}$	0.000	(0.000)	1.000	(0.000)
$rating_{it}$	0.014	(0.031)	1.014	(0.032)
$log(popularity_{it})$	-0.039***	(0.011)	0.962***	(0.010)
$games_i$	-0.115**	(0.041)	0.891**	(0.036)
$personal_apps_i$	-0.107*	(0.045)	0.898*	(0.041)
$utility_mobilespecific_i$	-0.140**	(0.045)	0.869**	(0.039)
$utility_mobileaccess_i$	-0.223***	(0.055)	0.800***	(0.044)
$learn_explore_i$	-0.069	(0.045)	0.933	(0.042)
$category_i$=content_consumption$_i$ (baseline)				
$below_1million_{it}$	-0.165***	(0.044)	0.848***	(0.037)
$5million_10million_{it}$	-0.125**	(0.040)	0.882**	(0.035)
$10million_50million_{it}$	0.052	(0.063)	1.054	(0.066)
$above_50million_{it}$	-0.169	(0.160)	0.844	(0.135)
$download_{it}$=1million_5million$_{it}$ (baseline)				
$dayssinceupdate_{it}$	-0.047***	(0.001)	0.954***	(0.001)
$screenshots_{it}$	0.001	(0.002)	1.001	(0.002)
$log(filesize_{it})$	0.110***	(0.016)	1.116***	(0.018)
$developer_appcount_{it}$	-0.000	(0.000)	1.000	(0.000)
Log Likelihood	-56098.79			
AIC	112263.6			
BIC	112592.4			
# of Obs.	156,903			
X^2	2626.29			
P	0.000			

*** p<0.001, ** p<0.01, * p<0.05; Clustered Standard Errors in parentheses.

Table A6. Tests for Pre-Trends

Panel A: Parallel Trend Assumption - Upgrading to Latest Version - PSM

Dependent Variable	(1) $rating_{it}$		(2) $log(popularity_{it})$	
$upgrade_group_i * time_{t-7}$	-0.001	(0.002)	0.015	(0.028)
$upgrade_group_i * time_{t-6}$	-0.001	(0.001)	0.003	(0.019)
$upgrade_group_i * time_{t-5}$	0.000	(0.001)	0.009	(0.021)
$upgrade_group_i * time_{t-4}$	-0.000	(0.001)	0.012	(0.019)
$upgrade_group_i * time_{t-3}$	0.002	(0.002)	-0.006	(0.019)
$upgrade_group_i * time_{t-2}$	0.002	(0.001)	-0.013	(0.017)
$upgrade_group_i * time_{t-1}$	-0.001	(0.001)	0.004	(0.017)
$upgrade_group_i * time_{t0}$	Omitted Baseline			
$upgrade_group_i * time_{t+1}$	0.001	(0.001)	0.003	(0.011)
$upgrade_group_i * time_{t+2}$	0.001	(0.001)	-0.000	(0.007)
$upgrade_group_i * time_{t+3}$	0.001***	(0.000)	0.005	(0.006)
$upgrade_group_i * time_{t+4}$	0.001***	(0.000)	0.004	(0.005)
$upgrade_group_i * time_{t+5}$	0.003***	(0.001)	-0.005	(0.009)
$log(popularity_{it})$	0.000	(0.001)		
$rating_{it}$			0.145	(0.169)
$\Delta rating_count_{it}$	0.000*	(0.000)	0.000*	(0.000)
$runtime_permissions_{it}$	-0.000	(0.000)	-0.001	(0.001)
$dayssinceupdate_{it}$	-0.000	(0.000)	0.000*	(0.000)
$screenshots_{it}$	-0.001	(0.001)	0.029	(0.026)
$log(filesize_{it})$	0.001	(0.002)	-0.031	(0.019)
$developer_appcount_{it}$	0.000	(0.000)	0.001	(0.001)
Constant	4.002***	(0.018)	-13.263***	(0.761)
Month Dummies	YES		YES	
App Fixed Effect	YES		YES	
Download Bucket Dummies	YES		YES	
F-test (pre-treat coef. jointly 0)	0.64		0.21	
p	0.73		0.98	
# of Obs.	53,983		53,983	
R-Squared	0.161		0.193	

*** p<0.001, ** p<0.01, * p<0.05
Clustered Standard Errors in parentheses

Panel B: Covariate Balance – LA-PSM Technique

Covariates	**Pre-Match (Overall)**	**Pre-Match (Treated)**	**Pre-Match (Control)**	**Difference**	**t-test**	**Post-Match (Treated)**	**Post-Match (Control)**	**Difference**	**t-test**
Rating Count	48410.14	52772.67	46468.36	-6304.31	-2.58	34854.14	35423.2	569.06	0.10
File Size	22.70	23.34	22.46	-0.88	-6.35	27.19	28.25	1.07	1.33
dayssinceupdate	189.65	108.17	219.05	110.88	99.71	88.10	91.37	3.27	0.99
developer_appcount	30.93	25.88	32.76	6.87	12.84	24.92	27.42	2.50	1.05
runtime_permissions	27.00	27.25	26.91	-0.34	-4.37	26.92	26.90	-0.02	-0.05
essential_permissions	3.93	3.12	4.22	034	32.69	2.06	2.03	-0.03	-0.15
nonessential_permissions	23.07	24.13	22.69	-1.48	-19.24	24.86	24.86	0.00	0.00
Rating	3.99	4.02	3.98	-0.04	-17.70	4.02	4.01	-0.01	-0.53
popularity	$6.31x10^{\wedge-6}$	$7.20x10^{\wedge-6}$	$5.98x10^{\wedge-6}$	$-1.22x10^{\wedge-6}$	-21.95	$6.09x10^{\wedge-6}$	$6.34x10^{\wedge-6}$	$0.25x10^{\wedge-6}$	1.19

Note: Apps that were already in Android 6.0 at the start of the panel are excluded from LA-PSM DiD analysis.

Panel C: Parallel Trend Assumption - Upgrading to Latest Version – LA-PSM

Dependent Variable	(1) $rating_{it}$		(2) $log(popularity_{it})$	
$upgrade_group_i * time_{t-7}$	-0.007*	(0.003)	-0.111*	(0.051)
$upgrade_group_i * time_{t-6}$	0.002	(0.003)	0.057	(0.032)
$upgrade_group_i * time_{t-5}$	-0.003	(0.002)	0.023	(0.030)
$upgrade_group_i * time_{t-4}$	0.000	(0.001)	-0.016	(0.034)
$upgrade_group_i * time_{t-3}$	-0.000	(0.002)	-0.021	(0.031)
$upgrade_group_i * time_{t-2}$	0.001	(0.002)	0.051	(0.031)
$upgrade_group_i * time_{t-1}$	-0.004	(0.002)	-0.042	(0.031)
$upgrade_group_i * time_{t0}$	(Omitted Base Case)			
$upgrade_group_i * time_{t+1}$	-0.001	(0.002)	0.003	(0.021)
$upgrade_group_i * time_{t+2}$	0.001	(0.001)	-0.010	(0.013)
$upgrade_group_i * time_{t+3}$	0.002**	(0.001)	-0.000	(0.009)
$upgrade_group_i * time_{t+4}$	0.001*	(0.000)	0.006	(0.008)
$upgrade_group_i * time_{t+5}$	0.001	(0.001)	0.006	(0.012)
$log(popularity_{it})$	0.001	(0.001)		
$rating_{it}$			0.339*	(0.170)
$\Delta rating_count_{it}$	0.000***	(0.000)	0.000***	(0.000)
$runtime_permissions_{it}$	-0.000	(0.000)	-0.001	(0.001)
$dayssinceupdate_{it}$	-0.000***	(0.000)	0.000	(0.000)
$screenshots_{it}$	0.004	(0.003)	0.016	(0.032)
$log(filesize_{it})$	0.003	(0.003)	-0.010	(0.032)
$developer_appcount_{it}$	0.000***	(0.000)	0.002	(0.001)
Constant	3.968***	(0.040)	-13.820***	(0.798)
Month Dummies	YES		YES	
App Fixed Effect	YES		YES	
Download Bucket Dummies	YES		YES	
F-test (pre-treat coef. jointly 0)	1.72		1.87	
p	0.1		0.07	
# of Obs.	29,324		29,324	
R-Squared	0.234		0.178	

*** p<0.001, ** p<0.01, * p<0.05; Clustered Standard Errors in parentheses

Table A7. Look-Ahead Propensity Score Matching Analysis of Effects of Delaying Upgrading

Dependent Variable	(1) DID – $rating_{it}$		(2) DID – $log(popularity_{it})$	
Model	3SLS		3SLS	
$\boldsymbol{upgrade_group_i * post_upgrade_t}$	0.004***	(0.001)	0.055***	(0.016)
$log(popularity_{it})$	0.001	(0.001)		
$rating_{it}$			-5.682**	(1.928)
$\Delta rating_count_{it}$	0.000***	(0.000)	0.000***	(0.000)
$runtime_permissions_{it}$	0.000	(0.000)	0.001	(0.000)
$dayssinceupdate_{it}$	-0.000***	(0.000)	-0.001***	(0.000)
$screenshots_{it}$	0.002**	(0.000)	0.014	(0.009)
$log(filesize_{it})$	0.006***	(0.001)	0.127***	(0.019)
$developer_appcount_{it}$	0.000***	(0.000)	0.000	(0.001)
Constant	3.247***	(0.014)	5.351	(6.232)
Count of Apps				
App Fixed Effect	YES		YES	
Month Dummies	YES		YES	
Download Bucket Dummies	YES		YES	
# of Obs.	61,652		61,652	
R-Squared	0. 988		0.652	

*** p<0.001, ** p<0.01, * p<0.05

Table A8. Collapsed DiD Analysis of Upgrading to Latest Version

Panel A: Analysis of Effects of Delaying Upgrade - PSM

Dependent Variable	(1) PSM DID – $rating_{it}$		(2) PSM DID – $log(popularity_{it})$	
	3SLS		3SLS	
$post_upgrade_t$	0.018***	(0.001)	-0.032	(0.054)
$upgrade_group_i * post_upgrade_t$	0.011***	(0.002)	0.116***	(0.033)
$late_upgrade_group_i * post_upgrade_t$	-0.016***	(0.004)	-0.479***	(0.041)
$log(popularity_{it})$	-0.010	(0.006)		
$rating_{it}$			0.363	(2.816)
$\Delta rating_count_{it}$	0.000***	(0.000)	0.000**	(0.000)
$runtime_permissions_{it}$	0.000*	(0.000)	0.001	(0.001)
$dayssinceupdate_{it}$	-0.000***	(0.000)	-0.001***	(0.000)
$screenshots_{it}$	0.002	(0.001)	-0.008	(0.016)
$log(filesize_{it})$	-0.004	(0.003)	0.086**	(0.032)
$developer_appcount_{it}$	0.000	(0.000)	-0.001*	(0.000)
Constant	4.123***	(0.080)	-13.099	(11.960)
App Fixed Effect	YES		YES	
Download Bucket Dummies	YES		YES	
# of Obs.	10016		10016	
R-Squared	0.993		0.860	

*** p<0.001, ** p<0.01, * p<0.05

Panel B: Analysis of Effects of Delaying Upgrade – LA-PSM

Dependent Variable	(1) PSM DID – $rating_{it}$		(2) PSM DID – $log(popularity_{it})$	
	3SLS		3SLS	
$post_upgrade_t$	0.032***	(0.002)	-0.068	(0.084)
$upgrade_group_i * post_upgrade_t$	0.004*	(0.002)	0.009	(0.020)
$log(popularity_{it})$	-0.005	(0.007)		
$rating_{it}$			4.453	(2.618)
$\Delta rating_count_{it}$	0.000***	(0.000)	0.000	(0.000)
$runtime_permissions_{it}$	0.000	(0.000)	-0.001	(0.001)
$dayssinceupdate_{it}$	-0.000***	(0.000)	-0.001**	(0.000)
$screenshots_{it}$	0.000	(0.002)	0.007	(0.016)
$log(filesize_{it})$	-0.000	(0.004)	0.080*	(0.038)
$developer_appcount_{it}$	0.000**	(0.000)	-0.005***	(0.001)
Constant	3.798***	(0.099)	-31.163**	(10.086)
App Fixed Effect	YES		YES	
Download Bucket Dummies	YES		YES	
# of Obs.	5,388		5,388	
R-Squared	0.991		0.856	

*** p<0.001, ** p<0.01, * p<0.05, $^+$ p<0.1

Table A9. Analysis of Effects of Delaying Upgrading to Latest Version – Bootstrapped SE

Panel A: Propensity Score Matching

Dependent Variable	PSM DID – $rating_{it}$		PSM DID – $log(popularity_{it})$	
	(1)		(2)	
upgrade_group$_i$*post_upgrade$_t$	0.009***	(0.001)	0.026**	(0.009)
late_upgrade_group$_i$*post_upgrade$_t$	-0.009***	(0.001)	-0.155***	(0.015)
log(popularity$_{it}$)	0.005***	(0.000)		
rating$_{it}$			1.464***	(0.059)
Δrating_count$_{it}$	0.000***	(0.000)	0.000**	(0.000)
runtime_permissions$_{it}$	0.000**	(0.000)	0.002***	(0.000)
dayssinceupdate$_{it}$	-0.000***	(0.000)	-0.001***	(0.000)
screenshots$_{it}$	0.001	(0.001)	-0.007	(0.007)
log(filesize$_{it}$)	0.002*	(0.001)	0.072***	(0.012)
developer_appcount$_{it}$	0.000***	(0.000)	-0.000	(0.000)
Constant	4.030***	(0.010)	-18.544***	(0.252)
App Fixed Effect	YES		YES	
Month Dummies	YES		YES	
Download Bucket Dummies	YES		YES	
# of Obs.	114,079		114,079	
R-Squared	0.991		0.722	

*** p<0.001, ** p<0.01, * p<0.05

Bootstrapped Standard Errors (500 replications) in parenthesis.

Panel B: Look-Ahead Propensity Score Matching

Dependent Variable	LAPSM DID – $rating_{it}$		LAPSM DID – $log(popularity_{it})$	
	(1)		(2)	
upgrade_group$_i$*post_upgrade$_t$	0.004***	(0.001)	0.030*	(0.013)
log(popularity$_{it}$)	0.005***	(0.000)		
rating$_{it}$			1.313***	(0.076)
Δrating_count$_{it}$	0.000***	(0.000)	0.000***	(0.000)
runtime_permissions$_{it}$	0.000	(0.000)	0.001	(0.000)
dayssinceupdate$_{it}$	-0.000***	(0.000)	-0.001***	(0.000)
screenshots$_{it}$	0.002*	(0.001)	0.003	(0.008)
log(filesize$_{it}$)	0.006***	(0.001)	0.083***	(0.015)
developer_appcount$_{it}$	0.000***	(0.000)	-0.002***	(0.000)
Constant	4.026***	(0.012)	-17.865***	(0.332)
App Fixed Effect	YES		YES	
Month Dummies	YES		YES	
Download Bucket Dummies	YES		YES	
# of Obs.	61,652		61,652	
R-Squared	0.988		0.713	

*** p<0.001, ** p<0.01, * p<0.05

Bootstrapped Standard Errors (500 replications) in parenthesis.

Table A10. PSM Analysis of Effects of Delaying Upgrading to Latest Version

Impact on Apps that do not reduce seeking nonessential permissions

Dependent Variable	(1) DID – rating$_{it}$		(2) DID – log(popularity$_{it}$)	
Model	3SLS		3SLS	
upgrade_group$_i$*post_upgrade$_t$	0.009***	(0.001)	0.156***	(0.034)
late_upgrade_group$_i$*post_upgrade$_t$	-0.009***	(0.001)	-0.253***	(0.039)
late_upgrade_group$_i$*post_upgrade$_t$*continue_overreaching$_i$	-0.001	(0.002)	-0.104**	(0.035)
log(popularity$_{it}$)	0.002***	(0.000)		
rating$_{it}$			-12.947***	(3.542)
Δrating_count$_{it}$	0.000***	(0.000)	0.000***	(0.000)
runtime_permissions$_{it}$	0.000***	(0.000)	0.004***	(0.001)
dayssinceupdate$_{it}$	-0.000***	(0.000)	-0.001***	(0.000)
screenshots$_{it}$	0.001*	(0.000)	0.006	(0.010)
log(filesize$_{it}$)	0.002**	(0.001)	0.101***	(0.015)
developer_appcount$_{it}$	0.000***	(0.000)	0.000	(0.000)
Constant	3.793***	(0.010)	35.149**	(13.343)
Count of Apps	5008 (2504 treated and 2504 control apps)			
App Fixed Effect	YES		YES	
Month Dummies	YES		YES	
Download Bucket Dummies	YES		YES	
# of Obs.	114,079		114,079	
R-Squared	0. 991		0.517	

*** p<0.001, ** p<0.01, * p<0.05

Appendix B

Machine learning approaches, specifically text analysis techniques, provide us with ways to derive insights and underlying motivations for apps' actions from unstructured data. In this investigation, we seek to uncover apps' motivations to delay upgrading to Android 6.0 using their text descriptions. Specifically, we employ a novel approach to Topic Modeling, which uses a Skip-Gram approach, popularized by Mikolov et al (2013) to learn Document-Topic and Topic-Word distributions. Once we determine the set of topics from the text description of apps, we employ Lasso Logit to determine which topics drive apps' decisions to delay upgrading to *runtime* permissions. Finally, we determine whether these topics correlate with apps' desire to over-seek permissions for ad-targeting purposes.

1. Neural Network-based Topic Modeling

Researchers have extensively used Topic Modeling, specifically Latent Dirichlet Allocation (LDA) models, in business research. Its applications include deriving business relatedness measures (Shi et al 2016), sentiments from users' comments (Khern-am-Nui et al 2018) or user proximity measures (Lee et al 2016). LDA model is an unsupervised technique that learns Document-Topic distributions as well as Topic-Word distributions and uses these distributions to determine the topics in each document. While LDA models are known to be effective in categorizing documents that coherently discuss a single topic, traditional LDA models have been shown to struggle when documents have overlapping topics or short descriptions with unique words (Hong and Davison 2010; Zhao et al 2011; Ho 2018). Hence, we employ a novel technique that uses a neural network to learn the topic-word and document-topic distributions (Ronnqvist 2018). Figure B1 shows the network structure.

The shallow neural network contains two parallel embedding layers. One of the layers is a document-topic embedding layer with d*t weights while the other layer is a word-topic embedding layer with w*t weights, where *d* is the count of documents, *w* is the count of words in the corpus and *t* is the count of topics, entered as a parameter. The ReLu-activated output of the two layers, a t*1 vector, passes through a dot operator followed by a sigmoid activation function. The network employs Skip-Gram model to learn

the weights by converting this unsupervised task to a supervised learning task. The training example is considered a positive sample if the word comes from the same document while an example is a negative sample if the word comes from a different document. In other words, if the word-document is labeled as a positive (or negative) sample, the training output is 1 (or 0). Since the weights of the two embedding layers are simultaneously identified, the backpropagation algorithm would push "similar" words and documents together, while pushing "dissimilar" words and documents apart in the vector space. Intuitively, if the word comes from the same document, the dot product should be closer to 1, whereas if the word comes from a different document, the dot product should be closer to 0.

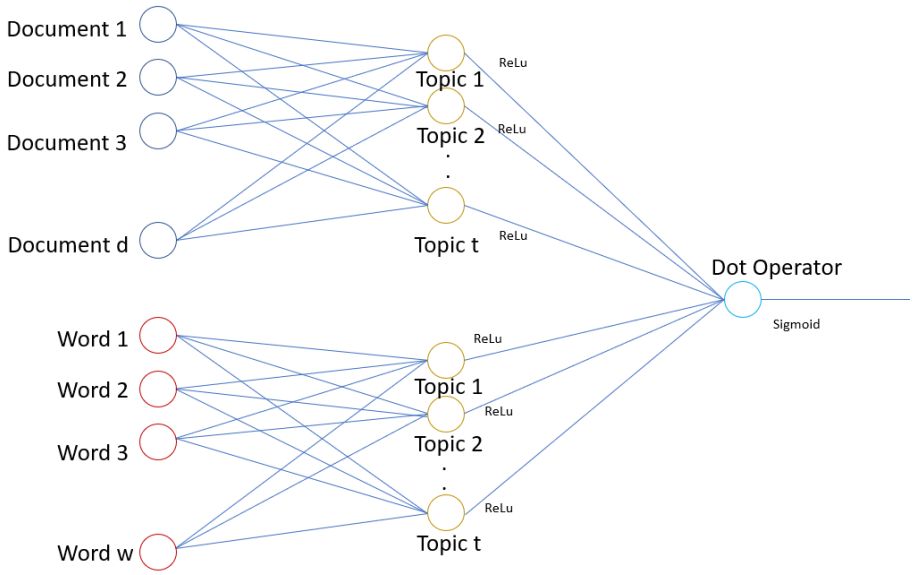


Figure B1: Neural Network Architecture

Since the sparsity of the document topic assignment is a desired feature (just as in LDA models), we use L1 regularization to the activated weights in both embedding layers to penalize non-zero trained weights in these embedding layers. This regularization is performed to push negligible weights closer to zero.

We compare the coherence values of the topics determined using this model with other popular models such as the LDA model and the LDA model with words weighted using TF-IDF. Figure 2 gives the plot of coherence values of the three models. We find that the UMass Coherence of Neural-Network based Topic Modeling is the lowest (lower value is better) compared to the other two techniques, giving confidence in our choice of model. From the graph, we find that the coherence value is the lowest (i.e., -3.38) when the

number of topics is set to 46. This is a justifiable number because Android broadly divided apps into 41 categories in 2016, with some categories being more generic than others, while some apps overlapped in their description. Furthermore, even if we select more topics, the Lasso procedure that we use in the next section would drop non-useful topics. Hence, we train our models to detect 46 topics and derive the document topic distribution for each document (i.e., text description). The text descriptions in the Play Store marginally change from month-to-month. Hence, to avoid over-fitting of our neural network model, we use three spaced out observations per app (i.e., descriptions of apps from the month of February 2016, 2017 and 2018).

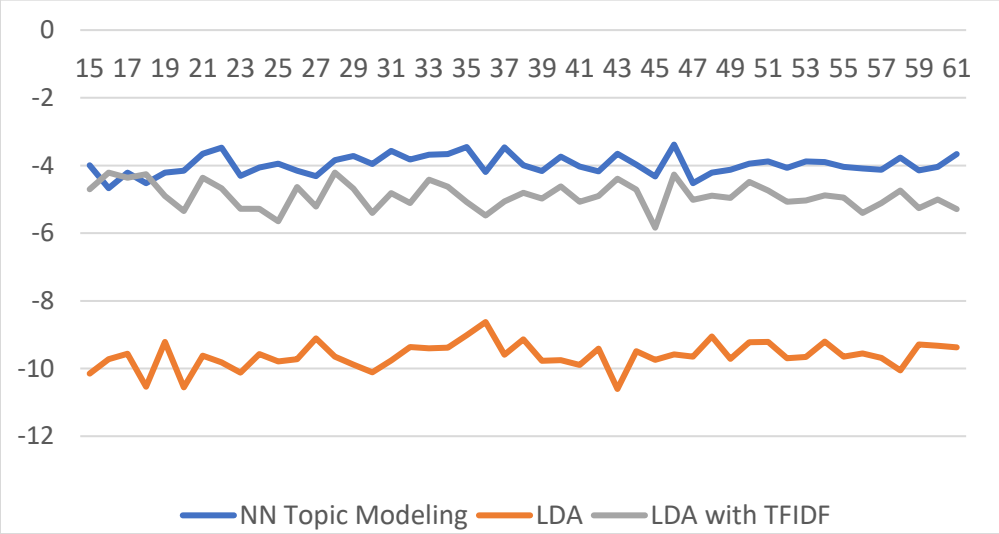


Figure B2: Coherence (UMass) plot comparing NN Topic modeling with LDA techniques

We train our model for 20 epochs. Finally, we extract the d*t matrix (the document embedding weights) to determine the topics for each of the document d.

2. Lasso Logit

The next task is to determine which of these topics are more likely to delay upgrading to the latest version of Android. Here we employ Lasso (Least Absolute Shrinkage and Selection Operator) regression technique, which has been used by Business Researchers to reduce the number of attributes that the model determines as insignificant for its performance (Ryzhov et al 2016; Sun et al 2016). Lasso is a technique that adds a penalty function to the MLE estimation that would shrink the coefficients of some of the

variables/features to zero if they do not add significantly to the model (Tibshirani 1996; Friedman et al 2010). Combining Lasso with logit models helps in classification/prediction tasks, in our case, predicting if apps upgrade to the latest version of Android. The penalized log-likelihood is :

$$\frac{1}{N}\sum_{i=1}^{N}\left\{y_i(\beta_0+\boldsymbol{x}_i'\boldsymbol{\beta})-\log\left(1+e^{(\beta_0+\boldsymbol{x}_i'\boldsymbol{\beta})}\right)\right\}-\frac{\lambda}{N}||\boldsymbol{\beta}||_1$$

where λ is the tunable parameter that we choose to determine the penalization. We determine the value of λ using two popular ways: 10-fold cross-validation and theory-driven (Belloni et al 2016). From 10-fold cross validation, the maximum value of λ for which the Mean Squared Prediction Error (MSPE) is within one standard error of minimal MSPE is 229.15. From Belloni et al. (2016), the value of λ that guarantees that the lasso logit is well-behaved is 261.43. For our analysis, we choose λ= 229.15.

Table B1: Selected Topics from Lasso Logit

Selected Topics	Logit Estimation	Post Estimation Logit
v0	1.009877	1.411916
v3	-0.23621	-0.55973
v5	-0.13831	-0.56271
v7	0.025635	0.338642
v8	0.396957	0.839866
v9	-0.60876	-0.97924
v10	0.988841	1.356154
v11	0.789203	1.183843
v12	-0.24428	-0.66776
v14	1.06233	1.36695
v15	1.321383	1.610648
v17	1.570454	2.065404
v20	-0.02175	-0.42808
v21	0.374641	0.656083
v22	-1.23143	-1.58886
v24	0.218238	0.579561
v27	0.316875	0.735328
v30	-0.14125	-0.45888
v32	-0.38861	-0.6747
v33	0.85119	1.197906
v34	-0.39628	-0.66701
v37	0.613943	0.918267
v38	-0.35076	-0.58373
v39	-0.57445	-0.82533
v40	-0.0646	-0.49499
v43	-0.00297	-0.29431
v45	0.100927	0.427575
Constant	-0.53957	-0.5556

Note: λ=229.15 for this model determined from 10-fold cross validation

Table B1 provides the list of 27 topics (out of 46) that significantly increase or decrease the propensity to delay the upgrade. Specifically, we find that 13 topics (out of 46) significantly increase the propensity to delay the upgrade while 14 topics decrease the propensity to delay the upgrade. Table B2 provides the topic-word distribution of these topics.

Table B2: Topic-Word Distribution
Panel A: Topic-Words that Predict Delay in Upgrading

Topic	Categorization	Example Apps	Topic-Words
5	Tools and Utility Apps	Flashlight AccuBattery	alert, light, battery, wifi, flash, fast, cell, gps, member, signal, led, internet, hotspots, speed, connection, monitor, network, flashlight, usage, data
12		The Weather Channel Weather Live Wallpaper	weather, forecast, wallpaper, live, clock, wind, temperature, radar, widget, rain, satellite, background, current, animation, condition, snow, hour, sky, location, pressure
30		CCleaner Cache Cleaner Super	cache, memory, protection, lock, battery, file, scanner, ram, clean, protect, antivirus, clear, data, anti, speed, storage, scanning, security, device, cleaner
34		ZEDGE Ringtones	ringtone, alarm, volume, timer, sound, set, recording, sleep, call, permission, sleeping, record, wake, clock, audio, incoming, mode, alarms, noise, voice
43		AppLock	lock, press, screen, backup, keyboard, galaxy, display, unlock, password, select, menu, widget, fixed, icon, button, calendar, note, off, bar, pattern
3	Apps for Children, Boys and Girls	Doctor Kids Operate Now: Hospital	tabtale, crazy, doctor, may, room, parent, downloading, bear, know, child, baby, puppies, kid, technical, puppy, them, term, plus, educational, interactive
9		My Talking Tom	child, kid, toddler, baby, educational, fun, animal, skill, talking, learning, letter, learn, old, memory, game, puzzle, ages, alphabet, drawing, coloring
32		Princess Salon	cat, fairy, christmas, pet, tree, winter, princess, dog, story, their, magic, hair, witch, girl, royal, santa, magical, dress, character, cute
22	Fashion Apps	Trendy Fashion Styles Dress Up	girl, fashion, dress, flower, clothes, beauty, designer, salon, wedding, outfit, makeup, have, hair, hairstyle, accessories, design, her, model, nail, looks
20	Communication Apps	Viber Messenger	call, sms, caller, calling, recipe, contact, chat, block, message, cake, blocked, voice, phone, number, voicemail, friends, messaging, sticker, send, don
38	Photo Apps	Draw it Ibis Paint X	drawing, draw, wallpaper, image, gallery, photo, card, size, frame, flip, color, picture, amazing, share, background, paper, creating, font, create, edit

39		Pixlr – Photo Editor PhotoGrid video & Pic Collage Maker	photo, camera, editing, picture, filter, editor, selfie, video, effect, instagram, frame, blur, collage, face, image, share, square, facebook, maker, text
40	Scriptures and other Reference Books Apps	King James Bible Allrecipes Dinner Spinner	bible, reading, recipe, prayer, dish, verse, along, restaurant, cooking, book, quran, holy, read, cafe, ingredient, ebook, food, last, god, islamic

Panel B: Topic-Words that Predict On-time Upgrade

Topic	Categorization	Example Apps	Topic-Words
0	Games Apps	Truck Simulator US	truck, site, car, gameloft, youtube, goo, license, advertisement, lights, city, contain, vehicle, zoo, social, ours, condition, agreement, party, transport, cargo
8		Jigsaw Puzzles Epic	ice, object, puzzle, hidden, link, studio, disney, term, advertising, privacy, brand, candy, rope, third, adventure, match, game, challenges, content, mini
21		PUBG Mobile	hero, battle, dragon, epic, enemy, rpg, fantasy, fight, monster, rare, kingdom, player, pvp, guild, wargame, strategy, item, quests, powerful, arena
24		EA Real Racing 3	racing, bike, car, race, track, performance, drift, point, bikes, races, ride, driver, cricket, upgrade, stunts, mode, speed, drag, buy, upgrades
10	Business Apps	Monster Job Search	job, property, sell, listing, find, estate, rent, search, sale, homes, logo, owner, buy, home, finding, copyright, near, looking, searches, price
27		Spotify Pandora	song, movie, radio, music, listen, story, artist, listening, shows, comic, watch, episode, anime, rock, video, library, playlist, voice, channel, title
33		Netflix Hulu	movie, subscription, episode, cinema, policy, trial, programming, watch, award, network, shows, film, privacy, may, direct, ticket, period, cartoon, package, access
37		Amtrak Fly Delta	train, flight, travel, booking, hotel, station, bus, route, trip, ticket, map, search, gps, timetable, book, destination, paris, price, cities, subway
45		Chase Mobile CreditWise from Capital One	bank, banking, recharge, transaction, balance, money, expense, account, credit, transfer, financial, payment, currency, mobile, payments, bill, card, bills, pay, loan
11	Physical and Mental Health Apps	Doctor on Demand Zocdoc: Find a Doctor	doctor, health, diary, magazine, period, resume, tips, award, calendar, professional, pregnancy, chart, website, medical, job, personalized, opera, record, industry, daily
17		Strava Keep Yoga	minecraft, workout, fitness, pocket, exercise, skin, training, track, day, weight, rest, yoga, abs, mojang, goal, healthy, water, trainer, activity, gym
7	Sports News Apps	ESPN FOX Sports	soccer, football, team, league, golf, player, season, nfl, spider, solitaire, baseball, club, basketball,

			field, cup, competition, stats, champions, sports, game
15		Cricbuzz	news, cricket, live, football, league, topic, bbc, sections, coverage, article, breaking, sport, latest, notification, team, sports, india, match, stay, global
14	Retailer Apps	Macy's JC Penney	shopping, deal, product, coupon, delivery, shop, gift, discount, barcode, store, order, pay, sales, price, buy, scan, brand, item, reward, list

3. Adware Analysis of Apps by Nexis Uni (LexisNexis Academia)

We want to investigate the impact of apps' strategic upgrade decisions on apps' advertising behavior. A simple means to study this is to investigate the extent to which the topics (specifically, the topic words) obtained in section 2 match with text documents that show apps' advertising motivation.

To obtain these text documents, we refer to media reports that cover "adware" (i.e., a piece of software that serves advertisement) on the Android platform. Specifically, we gather media coverage clippings that flag apps with aggressive adware problems. To obtain media coverage clippings, we use the Nexis (previously called LexisNexis) Database. We first obtained a list of all articles mentioning the keywords "adware" and "android" between January 2009 and October 2019. This search gives us 2,360 unique media coverage clippings. From this list of articles, we select 1,559 articles published in English.

Once we have the list of topics, we compare them with the word distribution of topics obtained above. Intuitively, if the topics from Panel A of Table B2 show greater similarity to these media coverage clippings than topics from Panel B of Table B2, we can infer that *in-app-advertising* significantly predicts the propensity to delay the upgrade.

3.1 Topic Level Analysis

We employ the cosine similarity method to measure the similarity between these topic words and the media coverage clippings. The cosine similarity technique gives a number between 0 and 1 based on the cosine distance (in all dimensions) between any two vectors that we wish to compare (we have outlined the process of converting text documents into vectors in the next paragraph). Intuitively, if two documents have a similar distribution of words, the cosine similarity measure between the vectors for these two documents

would be closer to 1. Similarly, if the documents have dissimilar word distribution, the cosine similarity measure would be closer to 0.

First, we create 27 text documents from topic words, one document for each of the 27 topics that we obtain from the Lasso technique in section 2. Next, we create a training dataset comprised of a total of 1586 documents, 1559 media report clippings and 27 topic documents. We employ a popular text mining technique called TF-IDF (Term Frequency – Inverse Document Frequency), which generates a vector for each of the text documents to compute similarity. Intuitively, the vector for any document is generated based on the frequency of words in that document and the importance of those words in all documents. The model trained this way gives us 1586 vectors; 1559 vectors for media report clippings, 13 for topics that predict delay in upgrade and 14 vectors for topics that predict on-time upgrade. We then obtain the cosine similarity measure between each of the 27 topics and the 1559 media report clippings (i.e., a total of 27*1559 = 42093 similarity measures).

Table B3 lists the top 25 cosine similarity values between topics and media report clipping. 24 out of the 25 top matches come from topics that predict delay in the upgrade.

Table B3: Similarity between Topic Words and Media Report Clippings

Rank	Topic	Cosine Similarity	Rank	Topic	Cosine Similarity
1	Delay Topic 39	0.262334	14	Delay Topic 30	0.164432
2	Delay Topic 40	0.260993	15	Delay Topic 30	0.16056
3	Delay Topic 30	0.216642	16	Delay Topic 30	0.160376
4	Delay Topic 38	0.214319	17	Delay Topic 43	0.15622
5	Delay Topic 30	0.19808	18	Delay Topic 30	0.155108
6	Delay Topic 30	0.184431	19	Delay Topic 30	0.147704
7	Delay Topic 30	0.18412	20	Delay Topic 12	0.147161
8	Delay Topic 30	0.183656	21	Delay Topic 30	0.147161
9	Delay Topic 39	0.176808	22	Delay Topic 5	0.145897
10	Delay Topic 39	0.174312	23	Delay Topic 9	0.13889
11	Delay Topic 30	0.170658	24	*On-time Topic 17*	*0.137926*
12	Delay Topic 30	0.168537	25	Delay Topic 12	0.137522
13	Delay Topic 30	0.167222			

Additionally, we find that 75 out of 100 of the top matches come from topics that predict delay in upgrade. This is despite the fact that there are only 13 topics that predict a delay in upgrading compared to

14 that predict an on-time upgrade. This clearly indicates that the topics that predict delay in upgrading to Android 6.0 are more similar to media clippings that report on adware in Android. In other words, the articles that discuss aggressive advertising on Android are significantly correlated with the topics that predict delay in upgrade.

3.2 Word Level Analysis

The key idea is to make word-level comparison between all articles on aggressive advertising in Android and topics that predict delay in/on-time upgrade. First, we create a single document from all 1,559 media report clippings. This single document contains 2,010,585 words. Next, there are 13 topics in Panel A of Table B2 (that predict delay in upgrade) and 14 topics in Panel B of Table B2 (that predict on-time upgrade). In order to balance the total number of words in both groups, we drop one of the topics with the lowest logit coefficients from panel B (Table B1 suggests that topic 7 has the lowest logit coefficient). From the chosen 13 topics, we select a unique set of words. Group A (chosen from Panel A, the group that predicts delay in app upgrade) has 235 unique words (out of 260), while Group B (chosen from Panel B, the group that predicts an on-time app upgrade) has 239 unique words.

We measure the frequency of occurrence of each of the words from Groups A and B in the single file made of all adware articles. We find that topic words from Group A, the group that predicts delay in app upgrade, are 124.3% as prevalent as topic words from Group B, the group that predicts an on-time app upgrade. Subsequently, we see that despite removing the five most occurring words from both groups, topic words from Group A are 122.88% as prevalent as topic words from Group B. This is despite the fact that the total unique words from Group A (235) are lesser than those in Group B (239). These clearly show a significantly larger correlation between the motivation to delay upgrading and the motivation to serve targeted advertising.

Next, we check for the number of times the runtime permissions are mentioned in these 1559 media report clippings. In Appendix Table A6, we had found that classes of permissions that predict a delay in the upgrade were permissions to access location, phone status and identity, microphone, phone memory, SMS and call logs. Analyzing the frequency of all permissions suggests that the above-mentioned six runtime

permissions are 320.38% as prevalent as other runtime permissions in these adware articles. This difference is significant and provides a clear indication that over-seeking these permissions are with an intention to serve targeted advertising.

To further strengthen our word-level analysis, we employ TF-IDF technique to determine the similarity between these topic words and the 1559 media coverage clippings. TF-IDF is a popular technique employed to generate vectors for documents based on the frequency of word occurrence in those documents. For this procedure, we train a TF-IDF model with a total of 1561 documents, 1559 documents from media coverage clippings and one document each for the two groups containing topic words. Once we obtain a vector for each document, we measure the cosine similarity between vectors for each of the 1559 articles, and the vectors that are obtained for Groups A and B. In total, we obtain 1559 similarity measures for each article with Group A and another set of 1559 similarity measures with Group B. We then compare the two groups of measures using the Student t-test. We find that the average similarity between newspaper articles and Group A is significantly ($t = 2.293$, $p < 0.05$) higher than the average similarity between newspaper articles and Group B.

Finally, we present anecdotal evidence to ascertain our text analysis techniques. Silverman (2018) reported that utility apps (system function improvement utilities) from a popular app developer with over two billion app downloads contained code for ad fraud. Similarly, Nohova (2018) from Avast finds that some weather apps contained aggressive adware. Proofpoint (2015) noted that hundreds of religious apps contained adware that compromised users' security. Goodin (2017) from Ars Technica reported that over 4,000 apps posing as communication and messenger apps recorded audio and call logs. Tee and Zhang (2019) from Symantec, O'Donnel (2019) from ThreatPost and Xu (2019) from Symantec reported that apps that pose as photography apps or fashion apps with millions of downloads carried adware to track user activities. Similarly, Arsene (2012) from BitDefender reported that apps that were primarily used by children (boys and girls) contained aggressive adware code.

In summary, this appendix shows that the topics derived from the text description of apps that we determine as the ones delaying upgrade correlate highly with app categories that are known to gather a lot of sensitive information from users with an intention to serve targeted advertising.

References

Arsene L (2012) Popular Kid-Friendly Android Apps in US Feature Aggressive Adware. *Bitdefender Hot For Security* (December 23), https://hotforsecurity.bitdefender.com/blog/popular-kid-friendly-android-apps-in-us-feature-aggressive-adware-4790.html

Belloni A, Chernozhukov V, Wei Y (2016) Post-selection inference for generalized linear models with many controls. *J. Business & Econom. Stat*, *34*(4), 606-619.

Friedman J Hastie T, Tibshirani R (2010) Regularization paths for generalized linear models via coordinate descent. *J. Statistical Software*, *33*(1), p.1.

Goodin (2017) Researchers report >4,000 apps that secretly record audio and steal logs. *Ars Technica* (August 12), https://arstechnica.com/information-technology/2017/08/android-users-bombarded-with-4000-spy-apps-three-land-in-google-play/

Ho G (2019) Why Latent Dirichlet Allocation Sucks. Retrieved on October 26. https://eigenfoo.xyz/lda-sucks/

Hong L, Davison BD (2010) Empirical study of topic modeling in twitter. *Proc. First Workshop on Social Media Analytics* . 80-88

Khern-am-nuai W, Kannan K, Ghasemkhani H (2018) Extrinsic versus intrinsic rewards for contributing reviews in an online platform. *Inform. Systems Res*. 29(4):871-92.

Lee GM, Qiu L, Whinston AB (2016) A friend like me: modeling network formation in a location-based social network. *J. Management Inform. Systems*. 33(4):1008-33.

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. *Advances in Neural Inform. Processing Sys.* 3111-3119.

Nohová A (2018) Weather, fitness, recipe and cryptocurrency apps spammed users with ads. *Avast* (May 16), https://blog.avast.com/suspicious-group-of-adware-apps-removed-from-the-google-play-store

O'Donnell L (2019) Adware-Ridden Apps in Google Play Infect 30 Million Android Users. *Threat Post* (April 24), https://threatpost.com/google-play-adware-30-million/144098/

Proofpoint (2015) Is nothing sacred? Risky mobile apps steal data and spy on users. *Proofpoint* (December 15), https://www.proofpoint.com/us/threat-insight/post/Risky-Mobile-Apps-Steal-Data

Ronnqvist S (2018) Neural Topic Modeling. Retrieved on 26 October 2019, https://github.com/sronnqvist/doc2topic

Ryzhov IO, Han B, Bradić J (2015) Cultivating disaster donors using data analytics. *Management Sci*. 62(3): 849-66.

Shi Z, Lee GM, Whinston AB (2016) Toward a Better Measure of Business Proximity: Topic Modeling for Industry Intelligence. *MIS Quart.*, *40*(4).

Silverman C (2018) These Hugely Popular Android Apps Have Been Committing Ad Fraud Behind Users' Backs. Buzzfeed News (November 26), https://www.buzzfeednews.com/article/craigsilverman/android-apps-cheetah-mobile-kika-kochava-ad-fraud

Sun T, Lu SF, Jin GZ (2016) Solving shortage in a priceless market: Insights from blood donation. *J. Health Econom*. 48: 149-65.

Tee MY, Zhang M (2019) More Hidden App Malware Found on Google Play with over 2.1 Million Downloads. *Symantec Threat intelligence* (September 23), https://www.symantec.com/blogs/threat-intelligence/hidden-adware-google-play

Tibshirani R (1996) Regression shrinkage and selection via the lasso. *J. Royal Statistical Soc.: Series B (Methodological), 58*(1), pp.267-288.

Xu E (2019) Adware Posing as 85 Photography and Gaming Apps on Google Play Installed Over 8 Million Times. *Trend Micro Security Intelligence Blog* (August 16), https://blog.trendmicro.com/trendlabs-security-intelligence/adware-posing-as-85-photography-and-gaming-apps-on-google-play-installed-over-8-million-times/

Zhao WX, Jiang J, Weng J, He J, Lim EP, Yan H, Li X (2011) Comparing twitter and traditional media using topic models. *European Conf. Inform. Retrieval.* 338-349

Appendix C

To determine when an app upgrades to Android 6.0 (or beyond), we installed the apps on emulators on research PCs, updated the apps on a monthly basis and observed the update status. A total of 16,705 of the most popular apps were installed on our emulators out of which 13,691 apps were in the dataset for at least 14 of the 24 months of the panel. Of the 13,691 apps, 2016 upgraded before April 2016, resulting in 11,675 apps retained in the Survival Models. From these 11,675 models, the propensity score matching (PSM) results in 2504 treated and 2504 control apps.

# Apps that were chosen to be installed on emulators	# Apps that were available on US Play Store to be installed	# Apps that were in the dataset for at least 14/24 months	#Apps that upgraded on or after April 2016 (Survival Analysis)	#Apps in retained after matching (DiD Analysis)
21,000	16,705	13,961	11,675	5008

Table C1: From the initial list of apps to actual apps used in the analysis

A key challenge with using the most popular apps in analysis is whether the inferences of this manuscript are generalizable. This appendix attempts to establish the external validity of the dataset by demonstrating that these 13,691 apps represent the global population of apps.

Understanding how the app distribution looks as compared to the global sample is the first step in establishing the external validity. The global sample is gathered from the Google Play Store's website for more than two million Android apps[29]. We present the findings in Table C2.

Table C2: Comparing the app distribution in our dataset with Global Sample
Panel A: General App Category Distribution

Categories	Our Sample	Global Sample
Tools	11.37%	10.23%
Entertainment	10.66%	11.70%
Personalization	8.13%	10.08%
Lifestyle	7.30%	8.54%
Photography	6.65%	3.88%
Education	5.02%	8.76%
Music and Audio	4.94%	5.71%
Productivity	4.92%	3.37%
Social	4.26%	2.74%
Books and References	4.21%	6.45%
Communication	4.11%	3.12%

[29] Apps with less than 10k downloads represent 0.0093% of all app installs and generally cater to a niche crowd (such as a school district, interest group), copycat apps, or are of inferior quality. These apps have been excluded from our analysis

Shopping	3.92%	2.07%
Finance	3.52%	2.99%
Travel and Local	3.37%	2.99%
Media and Video	3.31%	2.43%
News and Magazines	3.13%	3.02%
Health and Fitness	2.67%	3.20%
Sports	2.05%	2.16%
Business	1.98%	2.01%
Maps and Navigation	1.85%	1.93%
Weather	1.21%	0.86%
Comics	0.69%	0.40%
Medical	0.55%	1.05%
Libraries and demo	0.18%	0.29%

Panel B: Game App Category Distribution

Games Categories	Our Sample	Global Sample
Casual	19.88%	20.13%
Simulation	10.76%	9.55%
Action	9.50%	7.86%
Puzzle	9.42%	14.86%
Arcade	8.14%	11.10%
Racing	7.12%	5.63%
Educational	5.76%	4.85%
Sports	4.89%	3.65%
Strategy	4.10%	2.39%
Role Playing	4.04%	3.01%
Adventure	3.36%	4.13%
Casino	3.13%	2.81%
Card	2.51%	3.21%
Trivia	2.51%	2.56%
Word	2.18%	1.75%
Board	1.80%	1.77%
Music Games	0.90%	0.72%

As seen in panels A and B of Table C1, the distributions of apps in our sample is very similar to the Global distribution. This can be attributed to the large sample of apps that we chose in our dataset.

Next, another substantial concern regarding the generalization of our finding is whether the outcomes that are documented for the most popular apps would apply to less popular apps. While not having the upgrade data for less popular apps precludes us from conclusively demonstrating this, we attempt to understand how delaying upgrade impacts less popular apps within the dataset. For this, we divide the apps into 3 groups based on overall download categories: less than 1 million downloads (Low Download Group, $lowDG_i$), 1 million – 5 million downloads ($mediumDG_i$) and greater than five million downloads

($highDG_i$)[30]. We first compare the outcomes for apps in *highDG* with those in *lowDG* and then extrapolate the findings to less popular apps that are not in our dataset.

We re-estimate models (4a) and (4b) by including a three-way interaction term that identifies the effect of being in these groups, i.e., $late_upgrade_group_i * post_upgrade_t * download_group_i$, wherein the baseline comparison group is the *highDG*. The point estimate provides the heterogenous impact of being in the download groups *mediumDG* and *lowDG*. The results are shown in Table C3.

Table C3. Analysis of Effects of Delaying Upgrading to Latest Version – Moderating effects of Download Groups

Dependent Variable	(1) PSM DID $rating_{it}$		(2) PSM DID $\log(popularity_{it})$	
	3SLS		3SLS	
$upgrade_group_i * post_upgrade_t$	0.009***	(0.001)	0.157***	(0.034)
$late_group_i * post_upgrade_t$	0.006	(0.004)	-0.061	(0.098)
$late_group_i * post_upgrade_t * highDG_i$ (baseline)				
$late_group_i * post_upgrade_t * mediumDG_i$	-0.013**	(0.004)	-0.220*	(0.109)
$late_group_i * post_upgrade_t * lowDG_i$	-0.042***	(0.005)	-0.426*	(0.187)
$log(popularity_{it})$	0.002***	(0.000)		
$rating_{it}$			-13.034***	(3.598)
$\Delta rating_count_{it}$	0.000***	(0.000)	0.000***	(0.000)
$runtime_permissions_{it}$	0.000***	(0.000)	0.003***	(0.001)
$dayssinceupdate_{it}$	-0.000***	(0.000)	-0.001***	(0.000)
$screenshots_{it}$	0.001*	(0.000)	0.007	(0.010)
$log(filesize_{it})$	0.002**	(0.001)	0.101***	(0.015)
$developer_appcount_{it}$	0.000***	(0.000)	0.000	(0.000)
Constant	4.232***	(0.010)	43.657**	(15.150)
Month Dummies	YES		YES	
App Fixed Effect	YES		YES	
Download Bucket Dummies	YES		YES	
# of Obs.	114,079		114,079	
R-Squared	0.991		0.515	

*** p<0.001, ** p<0.01, * p<0.05

We see that the negative impact of delaying upgrading to Android 6.0 on apps' popularity measure is significantly higher for the lower download groups (*lowDG* and *mediumDG*). Similarly, we find that the impact of delaying the upgrade on apps' ratings is also the most conservative for apps in *highDG*, followed by apps in *mediumDG* and subsequently those in *lowDG*. We extrapolate this finding to argue that the negative impact of delaying upgrading on apps' rating as well as apps' permission seeking behavior only

[30] An app might move within groups over 24 months. We categorize such apps into the group where they spent most of their time in. e.g., spending 8 months in *lowDG* and 16 months in *mediumDG* puts an app in *mediumDG* category

exacerbates as we include apps with lower downloads. This gives confidence that our finding can be generalized for apps with lower downloads.

In summary, this appendix attempts to argue that the external validity of our finding from 13,691 apps is high and that the results are generalizable.

Appendix D
Alternative Explanations

Figure D1 graphically depicts the major changes that accompanied Android version 6.0. The only new restriction introduced in version 6.0 is the restriction to how consumers' sensitive information would be accessed by apps. Hence, we do not expect another platform-level change to trigger strategic behavior among a large sub-group of apps.

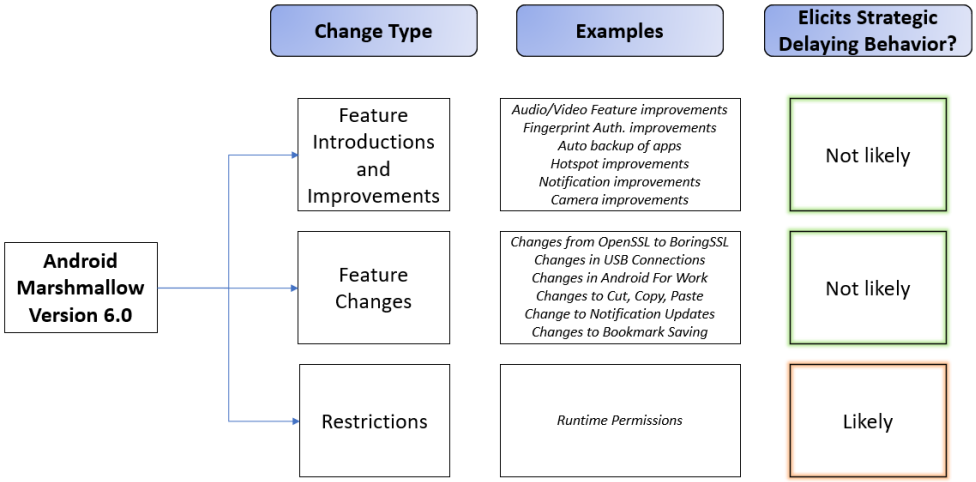


Figure D1: Changes introduced in Android 6.0

A potential reason for delaying the upgrade may also be driven by demand side reasons (e.g., the lack of demand for an upgrade from the existing users), supply side lack of benefits reason, misclassification of essential and nonessential permissions, extreme values of apps or other apps where consumers don't have a choice but to install the apps.

D1. Demand side Motivation Reasons: Apps may have low incentive to upgrade if they cater to a majority of users that will not upgrade their phones to the latest Android version. While such a scenario (existing users not upgrading) is feasible, Android's platform strategy provides good reasons why those apps should still upgrade. First, if a smart phone user has not updated the phone to version 6.0 or above, the app will get all permissions from the user during download, not individually during runtime. This is immaterial of whether apps have upgraded to Android 6.0 or not. Second, Android maintains an "Android Quality Score"

(AQI) for apps in the Play Store[31] and explicitly states that the quality score, and thereby app's Play Store ranking may be penalized by delaying the upgrade. Better Play Store rank aids in better discovery when new users employ the search bar or scroll through the "similar apps" list. Therefore, an upgraded app would benefit from attracting newer users because of a better Play Store ranking while continuing to get all data from old phones that have not upgraded.

D2. Supply Side Lack-of-Benefits: Apps that do not gain from new features in Android 6.0 will have few incentives to upgrade. Alternatively put, only apps that find new Android features more beneficial are more likely to upgrade while others do not invest effort to upgrade. A survey of all changes in the 6.0-version of Android (see Panels B and C of Appendix Table A1) informs us that all new features and improvements are beneficial to all apps equally. For example, Android made significant improvements to its notification services, which would equally improve the user notification effectiveness for all apps. Alternatively, the penalty that Android imposes to AQI for not upgrading should motivate apps to avoid negative consequences and therefore upgrade on time. Hence, we are confident that lack-of-benefits won't significantly impact apps' strategies for delaying this upgrade.

D3. Driven by misclassification of *essential* and *nonessential* Permissions: The study relies on a statistical technique that is used both in academia and industry (e.g., Peddinti et al. 2019; Taylor and Martinovic 2016) which uses a cutoff-voting method (8/10 apps should seek a permission for it to be *essential* for the peer group). The key goal of this technique is to flag unusual permissions amongst similar apps. We mitigate the concern of misclassification by trying various cutoffs for the polling. We re-estimated the model by changing the threshold to 70% (7/10 apps) and 90%. We find qualitatively similar results (presented in Columns (1) and (2) of Appendix Table D1). [32]

[31] https://developer.android.com/docs/quality-guidelines/core-app-quality

[32] Cutoff polling method may result in misclassification (e.g., unusual permissions sought by an innovative app being tagged as *nonessential*). However, immaterial of whether the method classifies such unusual permissions as *nonessential,* apps won't find it difficult to seek permissions essential for transactions (Zimmer et al. 2010). So, what is tagged as *nonessential* in this case won't affect apps decision to delay upgrade. Therefore, any potential misclassification (of unusual yet *essential* permissions as *nonessential*) will reduce the point estimate of negative effect of *nonessential* permissions, making the finding conservative.

D4. Driven by apps with extreme values for Dependent Variables: In our dataset, over 99% of the apps seek 12 or lower runtime permissions. To ensure that our outcomes are not driven by apps that are in the top 1% in terms of the count of permissions sought, we re-estimate the model by dropping top 1% of the apps. The result (presented in column (3) of Appendix Table D1) is consistent.

D5. **Driven by apps that users "must" install to access important offline services:** If an app is important for users (e.g., control app for a thermostat, banking app, shopping app etc.) such apps may decide not to update on-time. We mitigate this concern by running three sub-sample analyses. In the first analysis, we excluded apps that are "feeder" apps for offline services (such as bank apps and shopping apps) by excluding the Mobile Access category group. In the second analysis, we exclude controller apps (such as the thermostat and security cameras) by excluding the Lifestyle category apps. In the third analysis, we excluded personal apps (such as dating and social networking) by excluding the Personal category group. Results (presented in columns (4) through (6) in Appendix Table D1) are consistent.

D6. **Driven by unobservable developer level heterogeneity**: There may be certain developer specific heterogeneity, unobservable by researchers, that may drive the outcomes. To mitigate such a possibility, we re-estimate the survival model with developer fixed effects. Table D2 presents the outcome of the estimation. The results are consistent with the main models. We interpret this result with caution since including dummies in non-linear models such as the survival models may result in biases (Allison 2002). We find that the results are consistent with Mixed Effects model and the Cox Proportional Hazard Models.

Table D1. Analysis of Upgrading to Latest Version of Android –Robustness Checks

Dependent variable - $upgrade_{it}$	(1)		(2)		(3)		(4)		(5)		(6)	
Model Variant	*70% Threshold*		*90% Threshold*		*Exclude top 1%*		*Exclude Feeder Apps*		*Exclude Controller Apps*		*Exclude Personal Apps*	
	Hazard Ratio		Hazard Ratio		Hazard Ratio		Hazard Ratio		Hazard Ratio		Hazard Ratio	
$essential_\ permissions_70pct_{it}$	1.001	(0.002)										
$nonessential_\ permissions_70pct_{it}$	0.991***	(0.001)										
$essential_\ permissions_90pct_{it}$			1.004	(0.003)								
$nonessential_\ permissions_90pct_{it}$			0.993***	(0.001)								
$essential_\ permissions_{it}$					1.004	(0.002)	1.005*	(0.002)	1.005*	(0.002)	1.008***	(0.002)
$nonessential_\ permissions_{it}$					0.992***	(0.001)	0.992***	(0.001)	0.992***	(0.001)	0.993***	(0.001)
$\Delta rating_count_{it}$	1.000	(0.000)	1.000	(0.000)	1.000	(0.000)	1.000	(0.000)	1.000	(0.000)	1.000	(0.000)
$rating_{it}$	0.977	(0.031)	0.980	(0.031)	0.979	(0.031)	0.946	(0.031)	0.969	(0.031)	0.976	(0.033)
$log(popularity_{it})$	0.978*	(0.011)	0.978*	(0.011)	0.982	(0.011)	0.980	(0.011)	0.979	(0.011)	0.976*	(0.012)
$games_i$	0.928	(0.038)	0.940	(0.038)	0.938	(0.038)	0.948	(0.039)	0.934	(0.038)	0.935	(0.039)
$personal_apps_i$	0.894*	(0.041)	0.888**	(0.041)	0.890*	(0.041)	0.891*	(0.041)	0.900*	(0.048)		
$utility_mobilespecific_i$	0.831***	(0.037)	0.822***	(0.037)	0.850***	(0.038)	0.828***	(0.037)	0.831***	(0.037)	0.829***	(0.037)
$utility_mobileaccess_i$	0.757***	(0.039)	0.765***	(0.040)	0.763***	(0.040)			0.757***	(0.039)	0.760***	(0.040)
$learn_explore_i$	0.958	(0.043)	0.973	(0.044)	0.957	(0.043)	0.967	(0.044)	0.963	(0.044)	0.962	(0.044)
$category_i = content_consumption_i$ *(baseline)*												
$download_{it} = 5million_10million_{it}$ *(baseline)*												
$dayssinceupdate_{it}$	0.954***	(0.001)	0.954***	(0.001)	0.954***	(0.001)	0.955***	(0.001)	0.955***	(0.001)	0.955***	(0.001)
$screenshots_{it}$	1.003	(0.002)	1.003	(0.002)	1.003	(0.002)	1.003	(0.003)	1.003	(0.002)	1.002	(0.003)
$log(filesize_{it})$	1.070***	(0.017)	1.067***	(0.017)	1.071***	(0.017)	1.058***	(0.017)	1.072***	(0.017)	1.071***	(0.018)
$developer_appcount_{it}$	1.000	(0.000)	1.000	(0.000)	1.000	(0.000)	1.000	(0.000)	1.000	(0.000)	1.001***	(0.000)
Count of Apps	11,675		11,675		11,561		10,964		11,206		10,202	
Download Bucker Dummy	YES		YES		YES		YES		YES		YES	
Log Likelihood	-56291.02		-56294.13		-55677.15		-51608.63		-53600.81		-48304.25	
AIC	112618		112624.3		111390.3		103251.3		107235.6		96640.49	
BIC	112797.4		112803.6		111569.5		103419.7		107404.3		96797.72	
# of Obs.	156,903		156,903		155,330		148,016		150,396		136,837	
X^2	2190.34		2194.84		2197.72		2038.46		2088.90		2065.43	
p	0.000		0.000		0.000		0.000		0.000		0.000	

*** p<0.001, ** p<0.01, * p<0.05; Clustered standard errors in parentheses.

Table D2. Additional Analysis of Upgrading to Latest Version of Android Cox Proportional Hazard Model with Developer Fixed Effects

Dependent variable - $upgrade_{it}$	(1) *Hazard Ratios*	
$essential_permissions_{it}$	1.032***	(0.006)
$nonessential_permissions_{it}$	0.995**	(0.004)
$\Delta rating_count_{it}$	1.000***	(0.000)
$rating_{it}$	1.762*	(0.393)
$log(popularity_{it})$	1.009	(0.034)
$games_i$	2.680**	(1.004)
$personal_apps_i$	1.000	(0.283)
$utility_mobilespecific_i$	1.146	(0.292)
$utility_mobileaccess_i$	1.079	(0.421)
$learn_explore_i$	1.705*	(0.458)
$category_i$=content_consumption$_i$ (baseline)		
$below_1million_{it}$	0.909	(0.124)
$5million_10million_{it}$	1.071	(0.117)
$10million_50million_{it}$	1.373	(0.259)
$above_50million_{it}$	3.162*	(1.548)
$download_{it}$=1million_5million$_{it}$ (baseline)		
$dayssinceupdate_{it}$	0.955***	(0.002)
$screenshots_{it}$	1.004	(0.011)
$log(filesize_{it})$	1.125	(0.085)
$developer_appcount_{it}$	1.001	(0.002)
Developer Fixed Effects	YES	
Log Likelihood	-46136.31	
AIC	93964.62	
BIC	101609.4	
# of Obs.	62,095	
X^2	5039264.23	
p	0.000	

*** p<0.001, ** p<0.01, * p<0.05; Clustered standard errors in parentheses. Developer fixed effects drops observations of 4,320 apps due to no variation in DV