Product Labels for Mobile Application Markets

Position Paper

Devdatta Akhawe, Matthew Finifter
University of California, Berkeley
{devdatta, finifter}@cs.berkeley.edu

***Abstract*—Mobile application markets thrive, yet they are held back from their full potential by the *information asymmetry* between application developers and application consumers. A consumer has no way to gauge the security or reliability of an application, and a developer has no way to differentiate his application from that of the competition based on these factors. We argue that the centralized nature of mobile application markets, such as Android Market and the iOS App Store, afford them a unique opportunity to gather and present information. We discuss a number of ideas to leverage this opportunity to eliminate information asymmetry, including third-party certifications, qualitative descriptions of attack surface, and aggregated usage data.**

I. Introduction

Modern mobile platforms rely on application *markets* for distributing applications to end-users. An efficient market requires *information symmetry*: an understanding amongst both buyers and sellers regarding the quality of the goods on offer [2]. Mobile application markets are patently asymmetric; while the developer knows the real quality of the application, the user has access only to noisy indicators such as reviews and permissions.

The consequence of this scenario is that software developers are unable to differentiate their offerings based on quality, reliability, or security. The ability to differentiate and position products drives innovation and enables targeting under-served niches [15], [11]. In the absence of differentiation, there is no incentive for developers to innovate and position their products in terms of quality, reliability or security.

Markets thrive when they reduce or eliminate information asymmetry [2]. In real-world markets, product labels such as certifications or quantitative properties enable information symmetry. For example, mandatory nutrition facts on packaged food have enabled the massive health-food market by allowing customers to understand exactly which products are more nutritious than others, and in exactly which ways [5], [9]. Safety certification processes for electric products enabled their consumerization by eliminating consumer fear that use of a new gadget may cause injury. Services like CarFax prevent the used car market from becoming a market for lemons by providing car buyers with an accurate vehicle history. In short, eliminating information asymmetries can actually *expand* the size of a market, making it better for both buyers and sellers.

Taking our cue from real-world markets, we advocate for *product labels* in mobile application markets as a mechanism for mitigating information asymmetry. When we refer to product labels, we mean an informational element that presents the user with pertinent data about the application (product) in a standard format. We observe that market owners (e.g., Apple or Google) are in a unique position to develop powerful new product labels specific to mobile markets, and create the opportunity for consumers to effectively utilize these labels as part of their decision-making process.

For a given platform, the application market is (to a reasonable approximation) the only source of application installs. This puts the market owner in a unique position to collect data, as well as design, develop and enforce labeling. In addition, the market owner also controls the device (e.g., iPhone or Android). Control over the device means that the market owner is in a position to modify the device software to enable even better data collection and labels, a point we will return to in Section II-C. With both Android Market and the iOS App Store witnessing over a billion application installs every month [12], the sheer scale of mobile application markets makes their singular data collection opportunity even more potent.

It is not only the case that it is *possible* for market owners to develop and enforce novel labeling requirements; we believe they are also economically incentivized to do so. As we argued before, labels increase market efficiency and grow the market. Since market owners receive a percentage of the total market revenue, it is in their interest to encourage effective labels.

In the rest of this work, we focus on product labels for *security*, but the ideas we present are applicable to labels for quality, reliability and privacy.

II. Proposed Labels

In this section, we explore product labels for mobile application markets guided by the design and efficacy of real world product labels. While labels for software, particularly software metrics, have been extensively studied by researchers, we focus on the novel opportunities provided by centralized application markets. These proposals are merely a starting point; we hope they encourage a wider discussion around strategies for reducing information asymmetry in mobile application markets.

Mobile application markets provide a distinct opportunity for effective labels. The market can insert labels right at the decision point, where the user decides whether to install an application. To encourage differentiation based on labels, the

marketplace can present alternative products that offer similar functionality but have different security labels. Further, the market can personalize the labels shown to a user according to the user's profile, administrator policies, and/or security requirements. For example, a teenager could rely on crowd-sourced labels while military employees could rely more on certifications and compliance labels.

A. Certification

Certification is widely used today, from physical certification of products (e.g., fireproofing) to professional certification (e.g., a pilot). Certification influences a consumer's belief about a specific property of a product. Mobile operating systems rely on permissions, which are a form of coarse certifications provided by the platform. While all applications are required to state the permissions they use, this list of permissions provides little actionable information for users [7].

A broader set of certificates can verify a number of desirable properties. For example, a useful certification would verify that an application accesses the phone's camera only when the application is in the foreground. Another certification may be that an application communicates with only a single domain.

Roesner et al. study properties of device and API access that are intuitive to users [17]. They propose redesigning current architectures to enforce these "user-driven access control" properties. In our case, we argue for such properties as a differentiator for the more trustworthy applications. Of course, the certification mechanism is flexible and supports arbitrary properties. We rely on the market to figure out the exact set of properties to certify. Based on the supply and demand for particular certificates, the properties certified can change over time.

Certification can verify machine verifiable properties [3] as well as complex properties provided by the developer [16]. We expect that the former will be cheaper and more scalable, while high-assurance environments (e.g., military) can rely on the latter.

B. Testing and Standards

In the physical world, testing determines suitability of a product or compliance to a contract. For example, the tensile test measures the tensile strength of a material, which is an accepted standard for identifying material suitability. In the software realm, the Payment Card Industry (PCI) standard lays down a number of requirements for all software handling payment data.

We envision the emergence of independent testing providers, who test products for their fitness of purpose. That is, the testers evaluate and report on the extent to which the application does what it purports to do. The current user comment and review system is a rudimentary version of such a system, but it lacks a rigorous notion of reputation or trustworthiness in user reviews. Because mobile market owners control users' complete experience in using the market and act as the go-between for users and application developers, they are in a unique position to formalize independent testing and establish economic incentives for independent testing. Techniques and standards for software testing already exist in the literature [10], [4].

C. Qualitative Analysis

Qualitative analysis characterizes behavior that is hard to measure via cardinal numbers. While lacking the rigor of numbers, qualitative metrics are useful in the real world. For example, consumers can differentiate between shoes that offer "low", "medium", or "high" amounts of support. We envision such qualitative metrics to be particularly useful in cases where their quantitative versions may not be directly actionable.

For example, we believe a measure of attack surface can be a useful product differentiator. While standard techniques for quantitatively measuring the attack surface exist [13], it is not clear that the numerical value of an application's attack surface is actionable information. Instead, we envision bucketing attack surface metrics for similar applications. The applications are then categorized as "high", "medium", or "low" attack surface, as appropriate.

Another opportunity for qualitative metrics is application permissions. As we noted earlier, current permission systems are product certificates and provide limited actionable data. We intend to investigate how best to assign a numerical score to the extent of permissions. Similar to attack surface, we can then provide qualitative metrics that compare the extent of permissions across similar applications. A concerned user can act upon this information, similar to how a concerned shopper pays more for the (qualitatively) "light" version of a food product.

As we noted earlier, the market owner (e.g., Apple) typically also controls the software on the end-user's device (e.g., iPhone). We believe this presents a unique opportunity for powerful qualitative analysis of application behavior. The application platform (e.g., iOS) can record and store data indicative of the application's behavior. Data collected might include the domains contacted, GPS requests made, contact requests made, and so on. A user who has already installed an application may view visualizations of the data collected by his instance of the installed application. For example, a user may be able to view a histogram of domains contacted or a graph showing the total number of contact requests over time. A user can use this data as an audit mechanism and uninstall applications that exhibit egregious behavior.

Furthermore, the market owner can sample across the behavioral data gathered by the installed base of an application in order to aggregate data regarding application behavior. [1] Users who have not yet installed the application can view visualizations of the aggregate data, a qualitative label shown at install time.

D. Quantitative Analysis

Quantitative metrics are expressed as cardinal numbers. Real-world markets have a long history of using numbers for

[1]Due to potential privacy concerns, we do not propose collecting any personally identifiable data, and we support presenting only (1) a user's own data to himself, and (2) aggregate data over a large installed base to all users.

comparing product quality, e.g., sweetness index and tensile strength. The wine industry provides an anecdote in favor of numbers over qualitative values:

> [Wine] critics found that when they attempted to encapsulate wine quality with a system of stars or simply descriptors such as *good*, *bad*, and maybe *ugly*, their opinions were unconvincing. But when they used numbers, shoppers worshipped their pronouncements. [14]

The complete control over the install/uninstall experience makes novel quantitative metrics possible. A simple metric is the percentage of users who abort installation after looking at the permissions of an application. Further, if a user's qualitative audit (Section II-C) of an application's behavior causes her to uninstall the application, this data is a useful indicator of security issues. The market can alert new users to the rate at which users uninstall an application after auditing its usage. This feedback-based metric utilizes crowdsourcing and the centralized nature of the application market to gather data for a metric that would have been impossible in previous (decentralized) software ecosystems. We believe there are other such feedback-based metrics worth considering as well, e.g., how often a user kills an application after viewing its battery usage, or deletes an application's locally-stored data after viewing its disk usage.

III. Concerns

With metrics generated by the user network, we must be careful to consider the potential for Sybil attacks [6]. A developer may try to game the market to make his application appear better than competing applications (e.g., by performing numerous uninstalls of a competing application). As a first line of defense, the market owner can include a clause in the developers' terms of service that stipulates that developers will not engage in such behavior. The owner can also use anomaly detection to flag suspicious activity. It is worth noting that such attacks are already a concern today (e.g., for user ratings of applications), and market operators are therefore actively combating them [1]. Defenses against Sybil attacks are an active area of research, with a number of recent successes [19], [18], [20]. As we develop our proposal, we can adapt these defenses for product labels.

Another natural concern is *how* and *how well* users will be able to consume the additional information that we propose to make available to them. Because a conglomeration of numbers can confuse users, it might be necessary to combine multiple metrics into a single cardinal number. We acknowledge that the additional information may not, in fact, prove useful to the average user, despite our best efforts.

We can address this problem by implementing prototypes of some or all of these ideas and performing user tests with these prototypes. Our belief is that we cannot know until we try, and our hope is that even if the common user cannot effectively consume the additional information, a small subset of tech-savvy users can. Even in such a case, feedback mechanisms like reviews allow advanced users to influence the behavior of other users. Another possibility is pursuing research into how to make the information more useful to a user based on a profile of how security-conscious or tech-savvy she is.

Crowdsourced security labels make sense only in the context of *widespread* malice, and they will therefore not be able to protect against a targeted attack (i.e., an *advanced persistent threat* (APT) attack). Users concerned about such attacks must instead rely on non-crowdsourced labels, like certifications and testing.

IV. Future work

Future work includes modifying a mobile application market to gather the necessary data and present the qualitative and quantitative information we have described. In parallel, we can conduct user studies to evaluate the effectiveness of the labels in terms of the change in user behavior (similar to studies of food product labels [8]).

For example, we may design A/B tests that present different groups with different sets of labels. If the relative popularity of similar applications differs amongst the different groups, this may indicate that the additional information is a factor in users' decision-making process. Alternatively, user interviews can reveal users' understanding (or lack thereof) of the new information available to them, and it can give insight into how they are using the information to make decisions about application security.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. We are also grateful to Prateek Saxena and Emily Kumpel for their feedback. This material is based upon work supported by a NSF Graduate Research Fellowship and by Intel through the ISTC for Secure Computing. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the views of the NSF or Intel.

References

[1] Spamming of app comments : Google groups, 2010. http://bit.ly/Iw4xCj.

[2] G.A. Akerlof. The Market for "Lemons": Quality Uncertainty and the Market Mechanism. *The Quarterly Journal of Economics*, pages 488–500, 1970.

[3] Identifying Malicious Behaviors in Android Applications using Permission Event Graphs, 2012. Under (anonymous) submission.

[4] A. Bertolino. Software testing research: Achievements, challenges, dreams. In *Future of Software Engineering, 2007. FOSE'07*, pages 85–103. IEEE, 2007.

[5] Julie A. Caswell and Eliza M. Mojduszka. Using informational labeling to influence the market for quality in food products. Working Papers 25989, Regional Research Project NE-165 Private Strategies, Public Policies, and Food System Performance, 1996.

[6] J. Douceur. The sybil attack. *Peer-to-peer Systems*, pages 251–260, 2002.

[7] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android Permissions: User Attention, Comprehension, and Behavior. Technical Report UCB/EECS-2012-26, University of California, Berkeley, 2012.

[8] G.I.J. Feunekes, I.A. Gortemaker, A.A. Willems, R. Lion, and M. Van Den Kommer. Front-of-pack nutrition labelling: testing effectiveness of different nutrition labelling formats front-of-pack in four european countries. *Appetite*, 50(1):57–70, 2008.

[9] E. Golan, F. Kuchler, L. Mitchell, C. Greene, and A. Jessup. Economics of food labeling. *Journal of Consumer Policy*, 24(2):117–184, 2001.

[10] Iso/iec 29119 software testing standard, Feb 2012. http://softwaretestingstandard.org/.

[11] K.J. Kennedy and M. Moore. *Going the distance: why some companies dominate and others fail*. Financial Times Prentice Hall books. Financial Times/Prentice Hall, 2003.

[12] Jason Kincaid. Android Market: 10 Billion Apps Served So Far, And Another 1 Billion Each Month, December 2011. http://is.gd/5a5rcn.

[13] P. Manadhata and J. Wing. An Attack Surface Metric. *Software Engineering, IEEE Transactions on*, (99):1–1, 2010.

[14] Leonard Mlodinow. *The Drunkard's Walk: How Randomness Rules Our Lives*, pages 130–134. Pantheon, New York, New York, 2008.

[15] G.A. Moore. *Dealing with Darwin: How Great Companies Innovate at Every Phase of Their Evolution*. Portfolio, 2008.

[16] G.C. Necula. Proof-carrying code. In *Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 106–119. ACM, 1997.

[17] Franziska Roesner, Tadayoshi Kohno, Alexander Moshchuk, Bryan Parno, Helen J. Wang, and Crispin Cowan. User-driven access control: Rethinking permission granting in modern operating systems. In *Security and Privacy (SP), 2011 IEEE Symposium on*. IEEE, 2012.

[18] Haifeng Yu. Sybil defenses via social networks: a tutorial and survey. *SIGACT News*, 42(3):80–101, October 2011.

[19] Haifeng Yu, Phillip B. Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: a near-optimal social network defense against sybil attacks. *IEEE/ACM Trans. Netw.*, 18(3):885–898, June 2010.

[20] Haifeng Yu, Chenwei Shi, Michael Kaminsky, Phillip B. Gibbons, and Feng Xiao. Dsybil: Optimal sybil-resistance for recommendation systems. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, SP '09, pages 283–298, Washington, DC, USA, 2009. IEEE Computer Society.