Security Outside the Garden:

Improving the Security of Mobile Application Distribution

Aneesh Agrawal, Andrew Bartow, and James Loving

Table of Contents

I. Executive Summary

We propose that the Federal Trade Commission (FTC) publish guidelines for mobile application distribution that increase mobile security by allowing users to install and securely use applications ("apps") from third-party app stores, known on traditional platforms as app repositories. Future Internet growth is expected to be "mobile first," with Internet access through traditional desktop and laptop devices being preceded, or entirely replaced, by access through mobile devices.[1] These mobile devices rely on app stores that allow users to choose and download new software for their devices. Therefore, we must ensure that this application distribution infrastructure is secure and that users are able to choose the level of risk they accept. The FTC has a unique ability to set this standard under existing legal precedent.[2]

Current mobile application distribution systems are insecure. The Apple mobile operating system (mOS), iOS, does not allow third-party app stores, but nonetheless third-party stores exist. These stores are illegitimate and insecure, negatively impacting user security. The Android mOS allows third-party app stores, but these stores often fail to provide adequate user security. Moreover, the application distribution model presents fundamental threats to user security where attackers can insert malicious code onto users' devices.

To mitigate these problems, we propose that the FTC publish guidelines that

1. require mOS vendors to allow users to install and use applications from third-party application repositories and to verify the developer and repository

[1] Ben Bajarin, "Why the Internet's Next Billion Users Will Be Mobile-Only," *TIME*, accessed October 28, 2015, http://time.com/3589909/internet-next-billion-mobile/.
[2] "Federal Trade Commission v. Wyndham Worldwide Corporation, United States District Court for the District of New Jersey," accessed November 21, 2015, https://www.ftc.gov/system/files/documents/cases/140407wyndhamopinion.pdf.

signatures of applications at installation time, including checking the repository signature against a user-controlled whitelist;

   a. making APIs for core device functionality available for the entirety of the ecosystem;

   b. publicly documenting an easy to use method for the creation of third-party application repositories; and

2. require application repositories under FTC jurisdiction to protect user security through reasonable security measures, including at a minimum application screening, signing of apps by developers and repositories, metadata signing, and transport encryption, and encourage other repositories to do the same.

These guidelines will increase user security by legitimizing third-party app stores and preventing many fundamental threats to application distribution security. By allowing third-party app stores, these stores will be encouraged to comply with security best practices, and developers will be able to more effectively respond to user feedback and security concerns. Users will be able to control their level of risk by choosing which app stores to trust, further increasing the ecosystem's security as market forces separate secure stores from insecure ones. Finally, by introducing simple technical measures currently used in parts of the desktop ecosystem, theoretical "man-in-the-middle" attacks that allow adversaries to insert malicious code on users' devices will no longer be possible.

Although some stakeholders may raise specific concerns, these guidelines will further the general interests of all stakeholders. These guidelines will concretely

improve user security and choice, which are interests common to users, mOS vendors, application developers, app store operators, and the FTC. While this proposal up-ends the current iOS model, it will legitimize the third-party stores that exist regardless of Apple policy. This will increase user security. Moreover, in both the Apple and Android ecosystems, there are existing security measures to limit the impact of malicious code. These guidelines add an additional layer of security onto these existing measures, address theoretical attacks not addressed by existing mechanisms, and allow users to choose appropriate levels of risk. This is in the best interest of the mobile ecosystem.

II. Introduction

Future Internet growth is expected to be predominantly “mobile first,” with Internet access through traditional desktop and laptop devices being preceded, or entirely replaced, by access through mobile devices.[1] These mobile devices rely on app stores that allow users to choose and download new software for their devices. We must ensure that this application distribution infrastructure is secure and users are able to choose the level of risk they accept. To achieve these goals, the Federal Trade Commission (FTC) should provide mobile application distribution guidelines that

1. require mOS vendors to allow users to install and use applications from third-party application repositories and to verify the developer and repository signatures of applications at installation time, including checking the repository signature against a user-controlled whitelist;
    a. making APIs for core device functionality available for the entirety of the ecosystem;
    b. publicly documenting an easy to use method for the creation of third-party application repositories; and
2. require application repositories under FTC jurisdiction to protect user security through reasonable security measures, including, at a minimum application screening, signing of apps by developers and repositories, metadata signing, and transport encryption, and encourage other repositories to do the same.

These guidelines result from an detailed analysis of the mobile application ecosystem. This analysis follow an application from development being completed to the app being downloaded by a user: the application signing process used to guarantee the app's integrity and authenticity, the screening techniques performed by an app store prior to accepting an application, and the transmission security measures taken to protect communication between developers, app stores, and users. Based on this examination, we develop a set of minimum security standards, which form the technical foundation for a set of proposed guidelines. We then introduce our proposal and analyze it, including probable reactions and counter-arguments from various stakeholders. As an appendix to this report, we include a draft of possible FTC guidelines on mobile application distribution security.

In introducing this proposal, we seek to secure systems that currently exist in insecure states. The iOS ecosystem bans third-party app stores, forcing such stores underground; their illegitimacy is a threat to user security. The Android ecosystem allows third-party app stores, but these stores often do not provide adequate security. We seek a state where third-party stores are both plentiful and secure. Moreover, specific measures of our proposal will lead to increased security over current systems, especially in the face of theoretical attacks on user-store and developer-store communications.

In delineating security, we adopt the definition introduced by the International Standards Organization (ISO), which divides computer security into three facets: confidentiality, preventing unauthorized disclosure of information; integrity, protecting the "accuracy and completeness" of information; and availability, the

accessibility and usefulness of user access.[3] These properties reflect essentials of a secure system; while designers can make trade-offs between each, a dearth of any one of the three equates to an insecure system. Importantly, this definition applies to the mobile ecosystem as readily as the traditional one, as this definition is information-centric as opposed to mechanism- or system-centric. Additionally, security is a measure of degrees; there is no such thing as perfect security, and it would be infeasible to implement due to diminishing returns. Hence, we will analyze several technical mechanisms that purport to increase security in order to determine which ones provide enough of a benefit relative to their costs.

[3] "ISO/IEC 27000" (International Standards Organization, January 15, 2014).

III. Technical Analysis

Introduction

Distributing software is a process with a rich history and a variety of implementations used in practice, but doing so securely is still an area where industry practice has yet to catch up to theory. Traditional desktop platforms, which emerged early in the history of the computing industry, focused on simply making software distribution possible and did not consider how to make this secure. Restricted by backwards compatibility, these platforms continue to have problems with malware. Mobile operating systems, unconstrained with compatibility, have been able to create more secure distribution systems, but there continues to be a gap between mobile app distribution security and the limits of research in this area. In particular, the lack of official third-party stores on iOS has caused non-legitimate stores with few safeguards to pop up, exposing users to malware. On Android, where third-party stores are part of the ecosystem, a lack of standards for these stores and strong integration with the operating system have allowed malware to spread through sideloading and alternative, non-reputable stores.[4] Even in the face of a ban and technical restrictions preventing on third-party stores as on iOS, it is clear that users will seek out and find ways to utilize third-party stores, so it is imperative to find a model that integrates app distribution from third-party securely into the mobile ecosystem.

Traditional Application Security

Traditional computing platforms have high levels of incidence and spread of malware, as compared to newer platforms, making it clear that software distribution

[4] "McAfee Labs Threats Report," February 2015. http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q4-2014.pdf.

systems must be carefully architectured to prevent malware dissemination. For example, Windows' historic lack of a secure application distribution system, although now somewhat mitigated by the Windows Store, especially affects the non-mobile computing environment, as 85% of all traditional computing devices run Windows and two-thirds of those devices run older versions of Windows that lack access to the Windows Store.[5] To compound this problem, non-mobile platforms generally do not require app signing before application installation or execution.[6] This does not give users any assurance that they have downloaded and installed the application they intended. When combined with the lack of an attractive, vetted source of applications, this forces users to seek applications through general web searches and to execute this unvetted code. These systemic security shortcomings shine through in malware proliferation statistics. McAfee Labs estimates that of the 350 million samples of malware worldwide, only around 6 million samples affect mobile operating systems. Combined with the huge surge in popularity and usage of mobile devices, giving them comparable volume of use as traditional systems, this demonstrates the powerful ability of better security models to stop malware proliferation.

Application Distribution Security Model

Application distribution follows an application from completed development through to user download. While the term may elsewhere be inferred to specify simply the transmission/downloading of an application, here it also includes the application being signed by its develop, analyzed by the app store prior to acceptance, and then finally transmitted to the user. This expansion elucidates several security flaws in the

[5] "Desktop PCs Operating System Market Share 2012-2015 | Statistic." *Statista*. Accessed November 11, 2015. http://www.statista.com/statistics/218089/global-market-share-of-windows-7/.
[6] Wilcox, Jeff. "Getting Started with Code Signing for under $100," n.d. http://www.jeff.wilcox.name/2010/02/codesigning101/.

current model that would not be seen by focusing purely on transmission, such as theoretical attacks exploiting the incomplete application signing model.

Application Signing

A mobile application distribution system system can easily use application signing ("app signing" or "code signing") to improve security by upholding authenticity and integrity: it ties applications back to trusted actors and ensures they have not been modified. Any set of minimum security standards should include the signing of applications. During signing, each app is signed with a secret to attest its validity by the owner of the secret. An attacker trying to modify or forge an app would not have the secret and could thus not produce a valid signature, which can be easily detected.

App signing provides authenticity: it ties applications back to a specific developer and app store and protects against malicious updates. App signing cannot distinguish between trustworthy and untrustworthy applications; users must depend on application metadata (such as user reviews and developer history) or other external factors to decide who to trust.[7] However, an application repository can leverage application signing to convey that an application has gone through their approval process. This would allow a user to trust an application inasmuch as they trust that repository's approval process. Apple currently leverages app signing in this manner.[8]

App signing additionally provides integrity: it ensures that the app has not been modified since submission by the developer and distribution by the app store operator. This protects against attacks where a malicious actor attempts to pass off a malicious application as an application from a reputable source, thus abusing a user's trust for a

[7] Justin Cappos et al., "A Look in the Mirror: Attacks on Package Managers," in *Proceedings of the 15th ACM Conference on Computer and Communications Security* (ACM, 2008), 565–74, http://dl.acm.org/citation.cfm?id=1455841.
[8]" iOS Security, iOS 9.0 or Later," n.d. https://www.apple.com/business/docs/iOS_Security_Guide.pdf.

given repository or developer, because it ensures that apps cannot be forged or tampered with.

Android and iOS use different app signing models that achieve the same goal. On Android, developers sign applications, but developer keys do not need to be signed by a central key; the OS will trust any valid signature. A worrying practice of the Amazon Appstore is that submitted applications are resigned by a key generated by Amazon, breaking the developer-to-user chain of trust by removing the developer signature.[9] This does impart integrity guarantees, but partly destroys the authenticity measures granted by app signing. Unfortunately, this cannot be mitigated technically, and should be disallowed by policy. On iOS, developers also sign applications, but the keys used for signing are part of a Public Key Infrastructure (PKI): All keys must be signed by Apple, and iOS will only install and execute applications that have been trusted by Apple in this way. This also applies to applications under development, which are signed in a way that does not require application store submission for installation but prevents wider distribution.[10,11]

Alternative installation and update mechanisms exist that allow for security systems at the app store, such as application analysis, to be ignored; an implementation of application signing where applications are permanently linked to their approving repository would alleviate this concern, but not all application repositories currently leverage application signing in this manner. Applications can be sideloaded onto mobile devices. Sideloading is where users obtain an application file

[9] Amazon, "Publishing Android Apps to the Amazon Appstore," n.d., https://developer.amazon.com/public/support/submitting-your-app/tech-docs/submitting-your-app
[10] "Signing Your Applications," *Android Developers*, accessed October 11, 2015, https://developer.android.com/tools/publishing/app-signing.html.
[11] "Launching Your App on Devices," accessed November 11, 2015, https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/LaunchingYourApponDevices/LaunchingYourApponDevices.html.

from somewhere other than a centralized repository (such as an email) and then install that application on their device.[12] Currently in the Android model, only developers sign applications. If repositories also sign applications, it would allow second-hand users - who receive apps from decentralized sources, such as being e-mailed an app by a friend - to verify the origin of the app. Furthermore, this would enable the development of a *whitelisting* mechanism that could enable users to ensure they only install applications approved by repositories they trust. Updates to applications may also be downloaded through alternative methods or pushed to the user through channels outside of the app store.[13] As compared to an update delivered through the app store, these update mechanisms greatly increase the attack surface, potentially introducing vulnerabilities, but also provide a much faster distribution path for security hotpatches and vulnerability fixes among other updates.

Application Analysis

Application analysis, the suite of manual and automated checks that app stores can perform on applications to check for security flaws and illicit content, catches a high enough proportion of malicious applications to sufficiently ensure a high level of security for a large number of users. Malicious developers, or attackers that compromise the development or submission process of a developer, include hidden functions inside their apps that threaten security. Automated static and runtime analysis of applications allows app stores to easily, cheaply, and effectively screen for malicious behaviors in a large number of applications, including searching for patterns that may not be easily detectable by humans through the use of techniques such as

[12] "Sideload APK," *Sideload APK*, accessed November 10, 2015, http://sideloadapk.com/.
[13] "CodePush," *Microsoft Github*, accessed November 10, 2015, https://microsoft.github.io/code-push/.

machine learning.[14,15] Attackers can evade application app analysis by obscuring malicious behavior from automated analysis systems, but as analysis systems mature, these attacks become more difficult.[16] Manual review can also help screen for objectionable behavior and attributes not reviewable by computers.[17] The Google Play store, which uses app analysis as a primary means of screening applications, suffers from a malware intrusion rate of around one in every thousand applications, compared to other less reputable stores that perform little or no analysis and have up to a 33% malware penetration rate.[18]

App analysis currently fills needs that cannot be replaced by other mechanisms. Apple especially depends on app analysis to protect security due to iOS limitations involving private APIs, which are not intended for application access for security reasons. Due to the architecture of Objective-C, the language on which iOS is built, it is not possible to outright prevent these unauthorized accesses by technical means as is possible on Android.[19] Objective-C depends on message passing to invoke APIs, allowing all components to send messages to all other components, so the most Apple can do to protect private APIs is obscure the names of these APIs. (In contrast, Android

[14] "Android and Security - Official Google Mobile Blog." Accessed November 11, 2015. http://googlemobile.blogspot.com/2012/02/android-and-security.html.
[15] "Google Android Security 2014 Report," accessed November 16, 2015, https://drive.google.com/file/d/0BzcBZYaOx4TaSmtta0ZDZjlpcnM/view?usp=sharing&usp=embed_facebook.
[16] Oberheide, Jon, and Charlie Miller. "Dissecting the Android Bouncer," n.d. https://jon.oberheide.org/files/summercon12-bouncer.pdf.
[17] Perez, Sarah. "App Submissions On Google Play Now Reviewed By Staff, Will Include Age-Based Ratings." *TechCrunch*. Accessed November 16, 2015. http://social.techcrunch.com/2015/03/17/app-submissions-on-google-play-now-reviewed-by-staff-will-include-age-based-ratings/.
[18] "Report: 97% Of Mobile Malware Is On Android. This Is The Easy Way You Stay Safe." *Forbes*. Accessed November 11, 2015. http://www.forbes.com/sites/gordonkelly/2014/03/24/report-97-of-mobile-malware-is-on-android-this-is-the-easy-way-you-stay-safe/.
[19] "Using Private iOS APIs." *b2cloud*. Accessed November 12, 2015. http://b2cloud.com.au/tutorial/using-private-ios-apis/.

apps run on top of a managed virtual machine that controls all access to low level functionality and policies all memory accesses.) Currently, Apple manages this issue by disallowing private APIs in third-party apps as a matter of policy,[20] and using app analysis to fairly successfully screen for private API usage.[21]

To ensure the security of the apps they distribute, app stores should disallow execution of certain kinds of high-risk downloaded code. Attackers can circumvent the approval process entirely by downloading code from outside sources after distribution, allowing them to distribute exploitive code without app store oversight.[22,23] The iOS and Android platforms currently have such prohibitions.[24,25]

Application Transmission

The use of encrypted transport protocols (such as TLS) and metadata signing is necessary to provide integrity, confidentiality, and authenticity for app distribution. App distribution mechanisms, and the infrastructure supporting them, provoke several security concerns. Internet connections used to download apps are by themselves insecure, but many of these issues may be prevented through usage of TLS.

Securing application distribution with TLS prevents an attacker from performing man-in-the-middle (MITM) attacks against app stores, a vulnerability introduced when third-party applications are allowed. Broadly, a MITM attack is

---

[20] Apple. "iOS Developer Program Information," n.d.
[21] SourceDNA. "iOS Apps Caught Using Private APIs," October 18, 2015. https://sourcedna.com/blog/20151018/ios-apps-using-private-apis.html.
[22] Brussee, Paul, and Johan Pouwelse. "Autonomous Smartphone Apps: Self-Compilation, Mutation, and Viral Spreading." *arXiv:1511.00444 [cs]*, November 2, 2015. http://arxiv.org/abs/1511.00444.
[23] Foresman, Chris. "Proof-of-Concept App Exploiting iOS Security Flaw Gets Researcher in Trouble with Apple." *Ars Technica*, November 8, 2011. http://arstechnica.com/apple/news/2011/11/safari-charlie-discovers-security-flaw-in-ios-gets-booted-from-dev-program.ars.
[24] Apple. "App Store Review Guidelines," n.d. https://developer.apple.com/app-store/review/guidelines/.
[25] Google. "Google Play Developer Program Policies," n.d. https://play.google.com/about/developer-content-policy.html.

executed by an attacker that is logically located between the user and some server the user connects to; the attacker intercepts traffic flowing in both directions and can record, modify, or delete it without either party's knowledge.[26] With this technique, an attacker can intercept a user's request to download a specified application and replace the application with malicious content. Current implementations of application signing alone does not prevent this type of MITM attack. Currently on Android, any signature is equally valid;[27] thus, an attacker need only to sign her malicious Trojan application before transmitting it to the victim. If the infrastructure underlying TLS is trusted, TLS will categorically prevent MITM attacks; otherwise, the attacker could exploit the TLS PKI to impersonate a server, rendering TLS's protections worthless. Unfortunately, doubts exist about the trustworthiness of the TLS infrastructure.[28]

In order to secure application transmission in the case TLS cannot be trusted, we recommend app stores implement metadata signing (referred to on traditional systems as repository signing), which ensures integrity and authenticity of *metadata*: information about available packages, including a list of packages and signatures[29] of each package bundle, which is used by mobile app stores to determine what apps or updates are available and to verify the integrity of app downloads. Metadata signing means the metadata a repository provides must be accompanied by a signature, similar in nature to the signatures used in application signing, checkable by the mobile

[26] "Man in the Middle Attack," *Veracode*, accessed November 10, 2015, https://www.veracode.com/security/man-middle-attack.
[27] "Signing Your Applications," *Android Developers*, accessed October 11, 2015, https://developer.android.com/tools/publishing/app-signing.html.
[28] Michael Alan Specter, "The Economics of Cryptographic Trust: Understanding Certificate Authorities" (Massachusetts Institute of Technology, forthcoming).
[29] Secure hashes are one-way functions that produce a unique output for an input, enabling a user to check the integrity of the inputted file. For more information, see Tim Fisher, "Cryptographic Hash Function," *About Tech*, accessed November 15, 2015, http://pcsupport.about.com/od/termsc/g/cryptographic-hash-function.htm.

device that attests the repository created the metadata, preventing an attacker from altering metadata on the fly. The Google Play store, the Apple App Store, and the Amazon Appstore are all are closed source and use transport encryption, preventing examination of their source code or their communication. This makes it impossible to determine how they handle package metadata, if they implement metadata signing, and if so, if they do so securely.

TLS provides a desirable security benefit, even in the presence of signed repositories: Attackers cannot see the content being transmitted between the user and the application repository. This will significantly increase data confidentiality - thus increasing the user's privacy - and also bolster their overall security, as denying attackers information on users' applications also denies them information on which to base further attacks. TLS additionally prevents replay attacks, where the attacker intercepts and retransmits a message to impersonate one of the parties.[30] Both the Apple App Store and the Google Play store currently use TLS to secure connections between customers and the server,[31] although this is a recent security improvement.[32, 33]

[30] "Replay Attacks," *Microsoft Developer Network*, accessed November 11, 2015, https://msdn.microsoft.com/en-us/library/aa738652(v=vs.110).aspx.
[31] Paul Ducklin, "Apple Finally Adopts HTTPS for the App Store," *Naked Security*, March 9, 2013, https://nakedsecurity.sophos.com/2013/03/09/apple-finally-adopts-https-for-the-app-store-here-is-why-it-matters/.
[32] Oberheide, Jon, and Charlie Miller. "DISSECTING THE ANDROID BOUNCER." presented at the SummerCon 2012, n.d. https://jon.oberheide.org/files/summercon12-bouncer.pdf.
[33] Paul Ducklin, "Apple Finally Adopts HTTPS for the App Store," *Naked Security*, March 9, 2013, https://nakedsecurity.sophos.com/2013/03/09/apple-finally-adopts-https-for-the-app-store-here-is-why-it-matters/.

IV. Minimum Security Standards

Based upon this analysis of technical factors, we provide the following set of security measures which provide the level of security necessary and sufficient for the mobile ecosystem:

1. All mobile devices must have application repository systems that allow for third-party repositories. This will legitimize third-party app stores, which will encourage their adherence to security best practices. This also entails ensuring that APIs for base device functionalities are available for apps in third-party stores, not just the first party store.
2. Repositories must make a reasonable effort to ensure that all applications they distribute are free of malicious behavior. This analysis may be conducted in an automated fashion. Screening for malicious behavior prevents many harmful applications from reaching the market.
3. Applications must be signed both by their developers and by the repository disturbing them. Mobile operating systems must allow users to whitelist trusted repositories. Mobile operating systems must verify the presence and validity of these signatures before installation and only allow installation of applications originating from trusted repositories. This will prevent attackers from tampering with applications (e.g., pushing a fraudulent and malicious update).
4. Mobile devices must verify that repository metadata, such as lists of available applications, are signed. This will prevent attackers from

introducing malicious applications for download, even in the absence of transport security.

5. Mobile devices and application repositories must support and utilize transport encryption technology (e.g., TLS). This will prevent attackers from serving users malicious applications and protect users' privacy, such as the applications they download.

App stores and mOSes may exceed these security requirements, but any measures beyond those listed may actually decrease security and thus warrant close analysis. Specifically, mOS vendors may take measures to prevent or eliminate third-party stores. This prohibition is not necessary to ensure security: Third-party app stores are fully capable of providing sufficient security. Moreover, even in the face of a ban, unauthorized third-party stores will arise, and these simulacrum stores will likely not provide an adequate level of security.

Case Studies

Third party app stores currently exist, and the two major platforms offer drastically differing implementations, which offers a chance to compare the security strengths of both approaches in practice.

Third-Party App Stores on iOS

On other platforms that prohibit third-party app stores, operating system vulnerabilities outside the application distribution system have rendered mobile OS vendors unable to effectively enforce their prohibitions, and the resulting unauthorized app stores are rich with malware. For example, Apple does not permit iOS app distribution outside of their App Store; however, rampant unauthorized application distribution still occurs on iOS outside of the App Store. Many examples

exist of popular third-party app stores that have circumvented Apple's restriction. The popular Cydia platform, which attracted 10 million weekly users in 2011, allows users to install applications, operating system tweaks, and themes not approved by Apple.[34] A service called Zuesmos, created by a fifteen year old developer, uses Apple's developer signing system to allow users to install any sort of unvetted application they wish for a small fee.[35] Kuaiyong, a store focused on distributing pirated versions of applications available from the Apple App Store, reached a reported five million users.[36]

Although these unauthorized stores clearly offer value to users by distributing applications that Apple will not, their laissez-faire stance on app approval has allowed malware to flourish. Cydia repositories do not have to implement any sort of app analysis, signing, or transport encryption.[37] A recent attack carried out through a Cydia repository successfully compromised credentials of 225,000 Apple IDs from users in 18 countries, including the United States; the attackers then used these credentials to facilitate piracy.[38] Kuaiyong was criticized by security researchers for its opaque

[34] Shapira, Ian. "Once the Hobby of Tech Geeks, iPhone Jailbreaking Now a Lucrative Industry." *The Washington Post*, April 1, 2011. https://www.washingtonpost.com/business/economy/once-the-hobby-of-tech-geeks-iphone-jailbreaking-now-a-lucrative-industry/2011/04/01/AFBJ0VpC_story.html.
[35] "Zeusmos: Features," n.d. http://zeusmos.com.
[36] "How Has Apple Not Killed This? Chinese Startup Enables iOS App Piracy Without a Jailbreak." *Tech in Asia*. Accessed November 22, 2015. https://www.techinasia.com/china-kuaiyong-apple-ios-app-piracy-no-jailbreak/.
[37] "How to Host a Cydia™ Repository - Jay Freeman (saurik)." Accessed November 21, 2015. http://www.saurik.com/id/7.
[38]Claud Xiao. "Keyraider: iOS Malware Steals Over 225,000 Apple Accounts to Create Free App Utopia," August 30, 2015. http://researchcenter.paloaltonetworks.com/2015/08/keyraider-ios-malware-steals-over-225000-apple-accounts-to-create-free-app-utopia/

behavior and its far reaching EULA that did not allow for any sort of independent analysis to ensure the integrity of the store itself or the applications distributed.[39]

Reputable Third-Party App Stores on Android

Some third parties stores, such as the Amazon Appstore on Android, implement security practices that rival first-party stores, while unauthorized app stores on platforms that prohibit such third-party app stores are ripe with malware, suggesting that legitimizing third-party app stores may present a path for improving security. The Amazon Appstore has not only been able to fill unique gaps in the market, such as becoming the first western app store to sell apps in China,[40] but also has preserved the same security standards as the Google Play store. Applications distributed via the Amazon Appstore utilize app signing and undergo a review process before acceptance. [41] As the Appstore itself is distributed via HTTPS, we assume Amazon also implements transport security. We cannot verify any of the other security measures in use to ensure the integrity of application metadata. Nonetheless, the Amazon Appstore has a low malware incidence rate similar to that of the Google Play store.[42]

Malware in Third Party Stores on Android

Unfortunately, third party app stores are not a panacea for security. Compliance by third party stores with our other minimum security measures is necessary to ensure the security of the application distribution system. Many third-party app stores

[39] "Kuaiyong: The Short Version." Accessed November 14, 2015. http://abad1dea.tumblr.com/post/39556388225/kuaiyong-the-short-version.
[40] "Amazon Appstore Opens in China, Leaps Final Hurdle Before Kindle Fire Launch." *Tech in Asia*. Accessed November 21, 2015. https://www.techinasia.com/amazon-opens-appstore-china/.
[41] Amazon. "Publishing Android Apps to the Amazon Appstore," n.d. https://developer.amazon.com/public/support/submitting-your-app/tech-docs/submitting-your-app.
[42] TrustGo. "TrustGo Q4 Mobile Mayhem Report 2012," n.d. http://www.trustgo.com/images/en-GB/trustgo_q4_mobile_mayhem.pdf.

outside the American market struggle with security issues,[43] possibly in part due to a lack of regulatory structure and non-compliance with standard security measures, especially app analysis. Security measures, like application signing by application repositories in conjunction with whitelisting, will help insulate American users from these threats. Furthermore, the development of a set of cohesive minimum security industry standards that can be adopted globally, such as those found in our proposal, may lead to security practices in all app stores in the long term.

Technical Limitations

These standards include the most recent research in the field, are feasible with current technology, and implementations of each have already been seen in some areas of mobile or traditional application distribution.[44,45,46,47,48] However, there is one known shortcoming in the security these standards provide, although it is not a good attack vector in practice and has some mitigations, making it fairly harmless. For a good user experience, repositories (or rather the public/private key pairs they use for signing) are identified by human-friendly names, much the same way the Domain Name System (DNS) matches human-friendly names to website addresses. However, this turns out to be nearly insurmountable to do so in a secure way that prevents

[43] TrustGo. "TrustGo Q4 Mobile Mayhem Report 2012," n.d. http://www.trustgo.com/images/en-GB/trustgo_q4_mobile_mayhem.pdf.
[44] Justin Cappos et al., "A Look in the Mirror: Attacks on Package Managers," in *Proceedings of the 15th ACM Conference on Computer and Communications Security* (ACM, 2008), 565–74, http://dl.acm.org/citation.cfm?id=1455841.
[45] Apple. "App Store Review Guidelines," n.d. https://developer.apple.com/app-store/review/guidelines/.
[46] "Using Private iOS APIs." *b2cloud*. Accessed November 12, 2015. http://b2cloud.com.au/tutorial/using-private-ios-apis/.
[47] Google. "Google Play Developer Program Policies," n.d. https://play.google.com/about/developer-content-policy.html.
[48] Oberheide, Jon, and Charlie Miller. "Dissecting the Android Bouncer," n.d. https://jon.oberheide.org/files/summercon12-bouncer.pdf.

spoofing.[49] The only solutions so far utilize blockchains[50] such as Namecoin, but these require each device to store large amounts of data and do a lot of processing to verify identities,[51] making them infeasible for mobile devices with limited storage, power, and processing budgets. Current systems attempt to solve this problem by tying repositories to domains/URLs, relying on DNS and TLS, and thus at least ensuring the same level of security found elsewhere in the industry. In practice, this is not easily exploitable: the use of Trust On First Use (TOFU)[52,53] means an attacker only gets one chance to attempt this attack vector, and other security layers (transport encryption) or architectural design choices (bootstrapping trust from the initial repository à la retrieving the Amazon Appstore from Google Play) make this difficult to exploit.

V. Policy

Proposal

We call upon the Federal Trade Commission (FTC) to produce mobile application distribution guidelines that

3. require mOS vendors to allow users to install and use applications from third-party application repositories and to verify the developer and repository

[49] "Names: Distributed, Secure, Human-Readable: Choose Two," October 20, 2001. http://web.archive.org/web/20011020191610/http://zooko.com/distnames.html.
[50] Blockchains are distributed databases that are sequentially updated and confirmed through the database's use. The most famous example is the Bitcoin cryptocurrency.
[51] CCCen, *An Overview of Secure Name Resolution*, 2013, https://www.youtube.com/watch?v=eOGezLjlzFU.
[52] Trust on First Use (TOFU): a technique that means an application will accept and remember the identity that any service it connects to presents, allowing it to detect further changes in identity (signaling a MITM attack), but that leaves the very first connection open to a MITM attack.
[53] Viktor Dukhovni, "Opportunistic Security: Some Protection Most of the Time," *RFC Editor*, December 2014, https://www.rfc-editor.org/rfc/rfc7435.txt.

signatures of applications at installation time, including checking the repository signature against a user-controlled whitelist;

    a. making APIs for core device functionality available for the entirety of the ecosystem;
    b. publicly documenting an easy to use method for the creation of third-party application repositories; and

4. require application repositories under FTC jurisdiction to protect user security through reasonable security measures, including, at a minimum application screening, signing of apps by developers and repositories, metadata signing, and transport encryption, and encourage other repositories to do the same.

By setting an industry standard of the minimum measures necessary to ensure security, this should increase user security by

1. legitimizing third-party app stores, which will encourage their compliance with security best practices and increase user choice of individual risk acceptance, as well as driving a virtuous cycle of application feedback and updates, and
2. increase the security of third-party app stores' applications and mitigate attacks on the current application distribution model

Sample text for these proposed guidelines can be found in Appendix A.

Authority

The principal authority in enforcing mobile application distribution guidelines is the Federal Trade Commission. Established by the Federal Trade Commission Act, the FTC is "empowered and directed to prevent … unfair methods of competition."[54] In addition to the traditional meaning, "unfair competition" has been historically interpreted to include insufficient security measures. In FTC v. Wyndham, the Wyndham hotel and resort company was held liable for not having implemented "readily available" and "reasonable and appropriate" computer security measures and thus acting unfairly to their customers.[55] We argue that the minimum security measures delineated previously satisfy both standards and therefore request the FTC use this authority in policing mobile application distribution. Some desktop systems already utilize all of these measures,[56] and mobile ecosystems employ most of them already, making them readily available. As based on the technical analysis, we find these measures to create a lower bound for minimum security, making them reasonable and appropriate.

Scope

Although we propose regulatory action by the United States, we recognize that, because all major mOS developers and app store operators are based in the US,[16] these regulations will have a global impact. Mobile device usage has somewhat saturated the U.S. market, but is rapidly rising as it becomes accessible in developing nations around the globe, and all users deserve secure computing experiences. By setting a pervasive

[54] *Federal Trade Commission Act*, 15 U.S.C §§ 41-58, n.d.
[55] "Federal Trade Commission v. Wyndham Worldwide Corporation, United States District Court for the District of New Jersey," accessed November 21, 2015, https://www.ftc.gov/system/files/documents/cases/140407wyndhamopinion.pdf.
[56] "Apt - Debian Wiki," accessed November 25, 2015, https://wiki.debian.org/Apt.

standard, United States policymakers will effect global change, which mOS vendors have hitherto failed to implement. Given the massive proliferation of mobile devices anticipated in developing countries,[57] the harms of the status quo internationally exceeds even the domestic harms; for example, many of the users of non-legitimate third-party stores on iOS are found in China.

However, various factors may limit the effective reach of our proposal. New international mOS vendors that are not subject to US law may not follow the lead of United States. Mobile OS vendors may choose to only comply with the regulations where required to by law, leading to a balkanization of the mobile ecosystem. Application stores and mobile operating systems internal to a company or group (i.e. not available to the public) fall outside the scope of the FTC's authority to protect the public interest. Ultimately, wide adoption of these policies globally is needed to ensure maximum effect. Still, FTC regulation can prevent further harm in the US and presents a solid step forward towards solving these problems globally.

Implementation

We propose that the Federal Trade Commission publish mobile application distribution guidelines on its website, in a manner similar to many other guidelines regulating Internet commerce; see Appendix A for sample language. This will ensure high visibility without appearing heavy-handed. To allow for mOS developer and vendor compliance with these guidelines, we suggest that the FTC introduce the guidelines with an "effective by" date six months beyond their publication date.

[57] "The Growth of the Global Mobile Internet Economy". *Boston Consulting Group Perspectives.* Accessed 13 October 2015.

Analysis & Reactions

App store operators and mOS vendors who follow these guidelines will be able to provide a secure application distribution system for their users. However, convincing the mobile ecosystem to follow these guidelines may be difficult, as the guidelines run contrary to their established practice and may be seen as against their interest. Thus, we analyze the viewpoints and possible counter-arguments of mOS vendors, users, application developers, app store operators, and governments.

Mobile Operating System Vendors

Mobile operating system vendors are likely to push back on these guidelines, considering them infeasible. Apple is likely to have many complaints, and Google will likely also object to some of the required changes, although both of these two main mOS vendors have emphasized security when designing their operating systems. Apple, in an uncharacteristically revealing[58] iOS security white paper, states that "Apple designed the iOS platform with security at its core… Every iOS device combines software, hardware, and services designed to work together for maximum security."[59] Google states that "securing an open platform requires a robust security architecture and rigorous security programs. Android was designed with multi-layered security that provides the flexibility required for an open platform, while providing protection for all users of the platform."[60]

Apple may also argue that whitelisting is too complicated for users to utilize, and so allowing third-party app stores will shatter the security of their platform.

[58] As discussed previously, Apple typically obfuscates its security practices, including hiding them behind Non-Disclosure Agreements.
[59] "iOS Security, iOS 9.0 or Later," n.d. https://www.apple.com/business/docs/iOS_Security_Guide.pdf.
[60] Google. "Android Security." Accessed November 15, 2015. https://source.android.com/security/#background.

However, as seen with third-party stores on Android, whitelisting is a viable technique: The client for F-Droid, another third-party store available on Android, has a built-in interface with simple toggle buttons to enable and disable F-Droid compatible repositories (Figure 1). The utilization of TOFU[61] means most users do not have to handle the confusing details of key management,[62] but more security-conscious viewers can easily verify the identity of a repository by examining its fingerprint (Figures 2, 3).[63] These models presented by third-party app stores already in use on the Android platform show that the user experience issues are tractable. A user-friendly whitelisting system will improve the security of mobile ecosystems: Even if attackers compromise TLS, they must forge a repository's signature to impersonate it, allowing users to securely delegate the job of vetting applications to app stores they trust.

[61] Viktor Dukhovni, "Opportunistic Security: Some Protection Most of the Time," *RFC Editor*, December 2014, https://www.rfc-editor.org/rfc/rfc7435.txt.
[62] Alma Whitten and J. Doug Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0.," in *Usenix Security*, vol. 1999, 1999, https://www.usenix.org/legacy/events/sec99/full_papers/whitten/whitten.ps.
[63] A fingerprint is a short, cryptographically-secure representation of the identity of a private key, which is what the identity of each repository is tied to. Fingerprints allows humans to quickly check that the identity of a system or repository matches the expected identity in a secure way, without comparing long encoded strings.

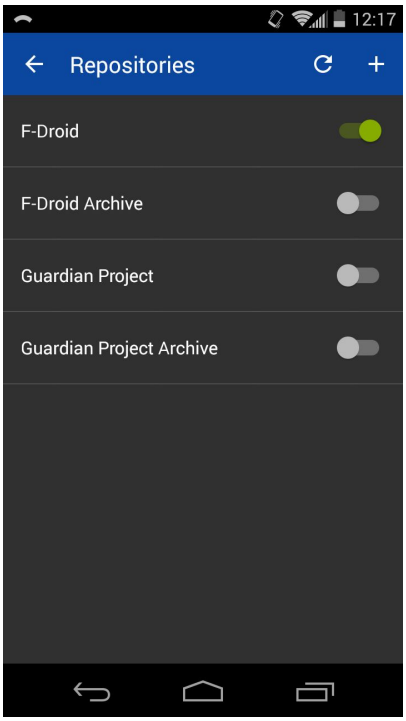

Figure 1: Whitelisting

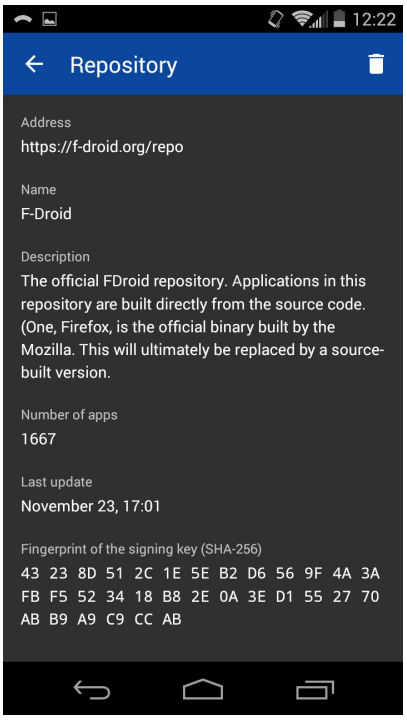

Figure 2: Repo Information

Figure 3: Repo Fingerprint

The proposed guidelines would integrate third-party store management into the operating system and allow for a better user experience (UX) through the use of integrated notifications. However, users may bypass a repository whitelisting system and sideload applications individually. Although there are clear benefits from using repositories to consume software including increased user control and choice, this is a real concern as evidenced by the prevalence of sideloaded apps on Android today. However, this threat is mitigated by the developer and repository signatures required by the guidelines, which allow some level of verification regardless of the application's installation vector.

This threat can be further mitigated by requiring users to explicitly enter the mOS settings beforehand to enable sideloaded apps and displaying a strong warning message whenever a user attempts to install a sideloaded app. Additionally, the user

would be warned if the application was not signed by a repository. This level of caution is justified, as the ease by which developers will be able to create third-party repositories will make sideloading apps far from normal.

Another concern limited to the iOS platform lies in Apple's current usage of static and dynamic app analysis to determine if apps attempt to access private APIs. In the long term, technical measures can mitigate this completely, but Apple will want to know how third-party stores can screen for private API usage in the near future. We offer two responses. First, the goal of allowing third-party app stores is that users will have the ability to gauge which stores they deem reputable and will keep them safe, letting them control their exposure to risk, instead of having the mOS vendor paternalistically decide for them. Second, Apple could expose the necessary parts of their automated static and dynamic analysis as a service which third-party stores could utilize (perhaps for a fee), allowing third-party stores to provide a more secure environment to attract users; market forces would then cause third-party stores to utilize this service to gain market share, so the bulk of users (with the lowest risk tolerance) will be protected.

Users

End users seek to use high quality, feature-filled apps, as well as assurance as to the security of their downloads, but often lack the technical background necessary to determine the safety of any given app and must rely on an app store operator's expertise to stay safe in the digital world. Users consider app descriptions and user reviews to assess the quality of apps before they purchase and install them.[64] Entire

[64] "Mobile Report: The App Store Factors Your Users Really Care about." *TUNE Blog*. Accessed November 15, 2015. http://www.tune.com/blog/mobile-report-the-app-store-factors-your-users-really-care-about/.

web sites exist dedicated to chronicling and seeking out the coolest new apps.[65]. Enterprise users, a subset of end users who use their devices for professional purposes, may also need the capability to install their companies' internal applications without distributing them in an app store.

By allowing third-party app stores and creating a whitelist system, these guidelines enable user choice and control over their risk exposure and experience. Extremely risk-averse user may choose not to enable any third-party repositories, while most users will take advantage of other stores that offer faster review times, a wider selection, or more thorough screening practices. Enterprises in particular will appreciate the ability to stand up their own repositories for internal applications and usage without having to go through external stores or having to procure special certificates.[66]

Application Developers

Enforcement of these guidelines will give application developers choice in app distribution platforms, which will further those developers' interests by freeing them from policies they find draconian and potentially driving up application revenue shares. Developers find the current state of security policies burdensome. The developer of Neato, an innovative note taking app for iOS, wrote "We were expecting Neato to be featured by Apple, but instead Apple … said that: 'You must remove the keyboard from the widget, or within 2 weeks your app will be removed from sale.'"[67]

---

[65] "iOS App Review Sites – 160 Sites For iPhone And iPad App Reviews." *iOS App Dev Libraries, Controls, Tutorials, Examples and Tools*. Accessed November 15, 2015. https://maniacdev.com/2012/05/ios-app-review-sites.

[66] Ellen Messmer, "Mobile Device Management: Apple's Extra Little Tricky Requirement," *Network World*, February 6, 2012, http://www.networkworld.com/article/2185540/smartphones/mobile-device-management--apple-s-extra-little-tricky-requirement.html.

[67] Neato. "Neato - The Fastest Way To Write Down Note on iPhone and iPad," n.d. http://neato.marblzz.com.

Complaints from notable developers about the sometimes capricious nature of the iOS App Store review process abound on Twitter.[68] Developers will also enjoy more leverage in obtaining a favorable revenue sharing agreement with app stores due to increased competition among app stores on revenue splits, or possibly may even stand up their own repositories to capture all revenue, empowered by easy access to documentation. On the Mac OS X platform, where a comparable system already exists, some developers already exercise these alternative options, with positive economic and innovative results.[69]

App Store Operators

This proposal will open up the app store market and bring increased competition, which will affect existing operators but overall increase the quality of the app store market. Existing stores are likely to be concerned about lost revenue from lower market share and downwards pressure on fees collected for service,[70] as well as the effects of increased competition in general causing the market to demand faster review times and more review transparency, caused by new operators that seek to differentiate themselves. New stores will appreciate the ability to enter a new market and compete on a more level playing field. Setting a precedent that stores will be open to potential liability if they don't keep abreast of security best practices will cause short-term compliance costs for stores to rise but improve the market in the long-term.

[68] Pierce, Greg. "Last Week's Rejection? Your Today Widget Does Too Much. This Week's? Your Today Widget Doesn't Do Enough. Seriously." Microblog. *@agiletortoise*, Invalid Date. https://twitter.com/agiletortoise/status/542391253757730817?ref_src=twsrc%5Etfw.
[69] "The 2014 Panic Report." *Panic Blog*. Accessed November 15, 2015. https://www.panic.com/blog/the-2014-panic-report/.
[70] Micah Singleton, "Rival Music Services Say Apple's App Store Pricing Is Anticompetitive," *The Verge*, May 6, 2015, http://www.theverge.com/2015/5/6/8558647/apple-ftc-spotify-app-store-antitrust.

Governments

Governments have an interest in promoting the security and vibrancy of the mobile application distribution system. In the United States, as illustrated in the Authority section above, this interest flows from the mission statement of the FTC.[71] By issuing these guidelines, the U.S. government will further the implementation of best security practices by mOS vendors and app stores designed to safeguard user security.

Concerns about Malicious Application Execution

Security advocates, be they a concerned member of the mobile user base, an engineer from a major mOS vendor, or a security researcher for an app store operator, may wonder what happens when a malicious app makes it through the application distribution system. Android and iOS, taking subtly different approaches, both provide measures that have succeeded at protecting users from the execution of malicious applications that slip through.

Both major mOSes rely on two main mechanisms: application sandboxing and permission management. Sandboxing limits the application's access to a "sandbox" that prevents it from accessing core system files or other applications; the permissions system allows users to specifically enable restricted functionality, such as an application accessing the device's GPS location data.[72] Together, these features uphold confidentiality and integrity by preventing malicious applications from accessing and modifying data they should not have access to, both system information and user/sensor data.

[71] Federal Trade Commission. "About the FTC." Accessed November 15, 2015. https://www.ftc.gov/about-ftc.
[72] William Enck, Machigar Ongtang, and Patrick McDaniel, "Understanding Android Security," *IEEE Security & Privacy* 7, no. 1 (2009): 50–57.

Android sandboxes applications through a series of mechanisms both at the level of the Linux kernel (which underlies Android) and in the Android OS itself. The Linux kernel provides the following guarantees:

1. Prevents [application] A from reading [application] B's files
2. Ensures that [application] A does not exhaust [application] B's memory
3. Ensures that [application] A does not exhaust [application] B's CPU resources
4. Ensures that [application] A does not exhaust [application] B's devices (e.g. telephony, GPS, bluetooth)[73]

Beyond the Linux kernel, Android enforces application security through programmatic restrictions on APIs that limit access to core services, and allow developers to create access methods for other applications (e.g., to allow a contacts application's data to be accessed by an e-mail application).[74] Users allow applications access to services and other applications through permissions, which can be classified as normal or dangerous. Users must agree to permissions at both levels when installing the application, but the application will re-prompt the user each time it wishes to utilize dangerous permissions. Dangerous permissions include accessing contacts and calendar events, sending/receiving SMS messages, and querying the device's sensors, including GPS, microphone, and camera.[75]

Apple's iOS does appear to effectively sandbox applications and utilize permissions, but we were forced to determine the details from a leaked Apple developer conference presentation.[76] Apple does not officially make information on iOS application sandboxing and permissions available to the public: This information

[73] "System and Kernel Security," *Android Open Source Project*, accessed November 11, 2015, https://source.android.com/devices/tech/security/overview/kernel-security.html.
[74] "Application Security," *Android Open Source Project*, accessed November 11, 2015, https://source.android.com/devices/tech/security/overview/app-security.html.
[75] "System Permissions," *Android Developers*, accessed November 11, 2015, https://developer.android.com/guide/topics/security/permissions.html.
[76] Ivan Krstic, "A Practical Guide to App Sandboxing," 2013, https://www.youtube.com/watch?v=5L4jjRlRjFI.

is hidden behind a paywall and a non-disclosure agreement and requires a paid Apple developer account for access.[77] Additionally, iOS allow users to specify applications' permissions, but do so on the fly;[78] this is equivalent to the "dangerous" Android permissions discussed previously.[79] This increases security, as a user will be more likely to deny a surprising permissions request than to deny a broad request with installation.

[77] "App Sandboxing," *Apple Developer*, accessed November 11, 2015, https://developer.apple.com/app-sandboxing/.
[78] "iOS Has App Permissions, Too: And They're Arguably Better Than Android's," *How-To Geek*, accessed November 11, 2015, http://www.howtogeek.com/177711/ios-has-app-permissions-too-and-theyre-arguably-better-than-androids/.
[79] "System Permissions," *Android Developers*, accessed November 11, 2015, https://developer.android.com/guide/topics/security/permissions.html.

VII. Conclusion

The adoption of these guidelines for mOS vendors and app store operators by the FTC will usher in concrete security improvements, the ability for developers to release updates quickly, and a new era of in which all users can choose who they trust and control their risk exposure. The proliferation of techniques such as application signing, application analysis, and security measures in application transmission will represent a real step forward in mobile security and make attacks against mobile devices more difficult. These guidelines will enable software vendors and consumers to iterate more quickly on new features, updates, security fixes and feedback, enhancing the security of application distribution and applications themselves. Beyond security, it may have a secondary effect of enhancing developers' capabilities to create novel applications. Users who had resorted to unauthorized app stores with lax security policies can migrate to sanctioned and more transparent systems that implement necessary security measures. Mobile devices owners, both end users and enterprises, can choose the risk level appropriate for their use case. Although no app store operator or mOS vendor currently implements the complete combination of minimum necessary security measures, the current major mobile operating systems and app stores already implement a subset of these guidelines, and the technology needed to comply with all of the guidelines exists today. The FTC can and should bring about a reality in which every mobile user enjoys the protection of a complete set of security measures.

Appendix A: Sample FTC Mobile Application Distribution Security Guidelines[80]

This security framework is intended to articulate best practices for companies involved in mobile application distribution. These best practices can be useful to companies as they develop and maintain processes and systems to operationalize mobile application distribution security practices within their businesses. To the extent the framework goes beyond existing legal requirements, the framework is not intended to serve as a template for law enforcement actions or regulations under laws currently enforced by the FTC.

Scope

Scope: The framework applies to all commercial entities that offer mobile operating systems or mobile application repositories ("app stores"). The framework does not apply to entities' internal distribution

Security through User Choice

Baseline Principle: Companies should enable and simplify user choice and increase user security by legitimizing users' possible choices.

A. The Substantive Principles

Principle: Mobile operating system vendors should incorporate security mechanisms in their products that empower user choice regarding risk and legitimize third party application stores.

B. Procedural Protections to Implement the Substantive Principles

Principle: Mobile operating system vendors should protect user security and empower user choice through the following:

1. Allow users to install and use applications from third-party app stores
2. Make Application Programming Interfaces (APIs) for core device functionality available to the entire ecosystem
3. Publicly document an easy-to-use method for the creation of third-party application repositories
4. Check and verify developer and repository signatures on apps at installation time, including checking the repository signature against a user-controlled whitelist

Security through System Design

Baseline Principle: Companies should promote consumer security throughout their organizations and at every stage of the development of their products and services.

A. The Substantive Principles

[80] This format is based on:
Federal Trade Commission. "Protecting Consumer Privacy in an Era of Rapid Change," March 2012.

Principle: Mobile application store operators should incorporate substantive security protections into their mobile application distribution practices.

B. Procedural Protections to Implement the Substantive Principles

Principle: Mobile application store operators should protect user security through security measures, including the following at a minimum:

1. Application screening
2. Enforcing application signing by developers and repositories
3. Metadata signing
4. Transport encryption between developers, users, and app stores

Promoting Self-Enforcement

The Federal Trade Commission is undertaking a project to develop a code of conduct for the distribution of mobile applications. The Commission will view adherence to such codes favorably in connection with its law enforcement work. The Commission will also continue to enforce the FTC Act to take action against companies that engage in unfair or deceptive practices, including the failure to abide by self-regulatory programs they join.

In all other areas, the Commission calls on individual companies, trade associations, and self-regulatory bodies to adopt the principles contained in the mobile application distribution security guidelines, to the extent they have not already done so. For its part, the FTC will focus its policy efforts on the two areas identified above, vigorously enforce existing laws, work with industry on self-regulation, and continue to target its education efforts on building awareness of existing mobile application distribution systems and use practices and the tools to control them.

Appendix B: Authors' Contributions and Acknowledgements

All authors contributed equally to this work. Aneesh Agrawal analyzed application and metadata signing. Andrew Bartow inspected the Apple ecosystem and existing third party app stores. James Loving evaluated application transmission and provided the initial policy proposal and analysis. All authors contributed to the policy proposal, analysis of stakeholders and counter-arguments, and review and revision.

The authors would like to thank Daniel J. Weitzner and Jessie M. Stickgold-Sarah for their insight and assistance in the revision process and Susan Perez for nutritional assistance.