Brian C. Rocca, S.B. #221576
brian.rocca@morganlewis.com
Sujal J. Shah, S.B. #215230
sujal.shah@morganlewis.com
Michelle Park Chiu, S.B. #248421
michelle.chiu@morganlewis.com
Minna Lo Naranjo, S.B. #259005
minna.naranjo@morganlewis.com
Rishi P. Satia, S.B. #301958
rishi.satia@morganlewis.com
MORGAN, LEWIS & BOCKIUS LLP
One Market, Spear Street Tower
San Francisco, CA 94105-1596
Telephone: (415) 442-1000

Neal Kumar Katyal, *pro hac vice*
neal.katyal@hoganlovells.com
Jessica L. Ellsworth, *pro hac vice*
jessica.ellsworth@hoganlovells.com
HOGAN LOVELLS US LLP
555 13th St. NW
Washington, D.C. 20001
Telephone: (202) 637-5600

Counsel for Defendants

Glenn D. Pomerantz, S.B. #112503
glenn.pomerantz@mto.com
Kuruvilla Olasa, S.B. #281509
kuruvilla.olasa@mto.com
MUNGER, TOLLES & OLSON LLP
350 South Grand Avenue, Fiftieth Floor
Los Angeles, CA 90071
Telephone: (213) 683-9100

Justin P. Raphael, S.B. #292380
justin.raphael@mto.com
Dane P. Shikman, S.B. #313656
dane.shikman@mto.com
Rebecca L. Sciarrino, S.B. #336729
rebecca.sciarrino@mto.com
MUNGER, TOLLES & OLSON LLP
560 Mission Street, Twenty Seventh Floor
San Francisco, CA 94105
Telephone: (415) 512-4000

Jonathan I. Kravis, *pro hac vice*
jonathan.kravis@mto.com
MUNGER, TOLLES & OLSON LLP
601 Massachusetts Ave. NW, Suite 500E
Washington, D.C. 20001
Telephone: (202) 220-1100

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

IN RE GOOGLE PLAY STORE ANTITRUST LITIGATION

THIS DOCUMENT RELATES TO:

Epic Games Inc. v. Google LLC et al., Case No. 3:20-cv-05671-JD

Case No. 3:21-md-02981-JD

DECLARATION OF EDWARD CUNNINGHAM IN SUPPORT OF GOOGLE'S PROFFER REGARDING EPIC'S PROPOSED REMEDIES

Judge: Hon. James Donato

DECLARATION OF EDWARD CUNNINGHAM

1. I am a Director of Product Management at Google, focusing on Android operating system improvements in the area of security, privacy and abuse. In particular, I have devoted a lot of time in recent years towards improvements to Android developer APIs, in order to better protect users from malware and other types of unwanted apps that have a negative security, privacy or user experience impact. In that role, I am responsible for working with engineers, designers and other team members to make these improvements to Android. The facts set forth herein are within my personal knowledge and if called as a witness, I could and would competently testify to them.

2. I have reviewed the portions of Epic's proposed injunction relating to Catalog Access, Library Porting, and Distribution of Third Party Stores through the Play Store. I offer this declaration to describe the technical implementation and anticipated resource allocation required for some of these remedies. This declaration reflects my current analysis within the short timeframe provided and based on the limited description of the remedies set forth in Epic's proposed injunction. If Google were ordered to implement these remedies, it is possible that Google could encounter unanticipated issues requiring different methods of implementation, which may entail a different resource burden and cost.

3. Because of my role, this declaration is focused on the changes that are needed to Android in order to accomplish some of these remedies.

Library Porting

4. In this section, I describe the changes to the Android operating system that would be necessary to implement the "library porting" remedy as described in the proposed injunction.

Unauthorized Cross-Store Updates & Android 14

Unauthorized Cross-Store Updates

5. Prior to Android 14, any preloaded app with the INSTALL_PACKAGES permission, which includes preloaded app stores, could update any app on the device without user permission or notification. For example, if the user had three preloaded app stores on the device, all three app stores could attempt to automatically update any app that the user installed from any app store. This phenomenon, where app stores push updates to apps that are expected to be

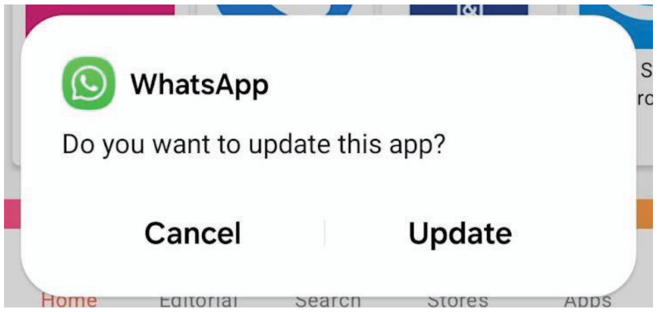
updated via a different app store, either inadvertently or intentionally, is sometimes referred to inside Google as "app clobbering." It led to a number of problems before Android 14.

6. First, unauthorized cross-store updates led to changes in app behavior that were unexpected to the user, app developer, and app store, particularly relating to in-app billing. When an app was installed from App Store A (with a version of the app where the developer had integrated App Store A's billing library), but was later automatically updated by App Store B (with a version using App Store B's billing library), the in-app purchase experience for that app would change without the user's knowledge. This could result in the loss of prior in-app purchases or purchased subscriptions. In other cases, this would simply result in a poor user experience (e.g., having to re-enter billing information), and frustration—and potentially revenue loss—on the part of the app developer and original app store (App Store A) developer.

7. Second, unauthorized cross-store updates led to unstable apps and crashes. In some cases, a store would silently push an update to an app that was installed by another store, which resulted in that app crashing or failing in some way. For example, in 2022 the Oppo App Store pushed an "update" to Google Chrome (which Google did not authorize) to Android users, but the version that the Oppo App Store pushed was only intended and tested for use on specific older versions of the Android operating system. As a result, for some users with devices running newer versions of the Android operating system, the "update" of Chrome by the Oppo App Store led to an inability to load webpages on Chrome.

8. Third, I understood the potential for developers to have difficulty executing staged rollouts of new app versions. App stores like Play offer developers ways to stage rollouts of new app versions, e.g. to first make the version available to a certain percentage of users, before going to 100% availability. That staged rollout could be based on various factors including percentage of user base, geo-targeting, specific groups of users, etc. This developer control could be rendered much less effective if another store pushes the app update to users who were otherwise not eligible for the new version according to the staged rollout without developer permission. This can interfere with experiments, or just result in a beta version of an app accidentally being made available to users who were not signed up to receive such a version.

9. By contrast, prior to Android 14, any user-installed app with the REQUEST_INSTALL_PACKAGES permission, which includes sideloaded app stores, required user confirmation in order to update an app on the device which had previously been installed by a different store. This user confirmation was obtained through an Android operating system dialog that looked like this:



WhatsApp

Do you want to update this app?

10. Because user confirmation for such updates was required, there was no risk of truly unauthorized cross-store updates originating from sideloaded app stores. Nonetheless, it remained possible for sideloaded app stores to prompt users to authorize such updates, and there was no guarantee that users would remember which store they had originally used to install the app (and thus would have expected updates to be delivered from). As such, users could authorize these cross-store updates without any reminder or understanding of the consequences. Furthermore, apps that users initially installed from sideloaded app stores were at risk of unauthorized updates from preloaded app stores (for which there was no equivalent user confirmation required, as I described above).

Android 14 & "Update Ownership"

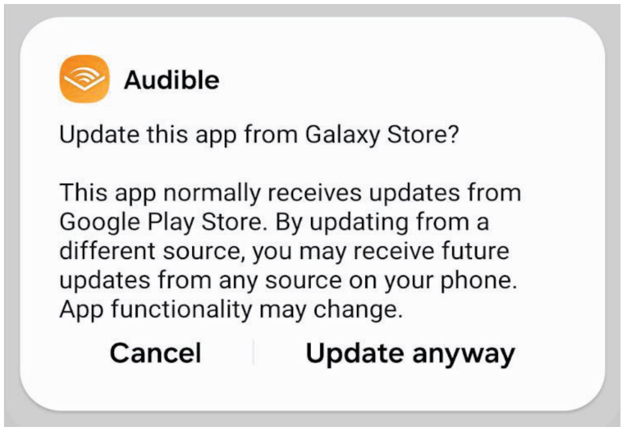
11. To address these problems, Android 14 (for which work began in earnest around May 2022, and which was ultimately released to the public in October 2023) introduced the concept of "update ownership." This gives any install source (be that an app store, or any other type of installer) the ability to require user confirmation before another install source can install updates to apps installed from that source. In other words, on devices running Android 14, any install source can now ensure that, by default, an app installed from that source will only receive automatic updates from the same source, unless and until the user decides otherwise. Google's

intention with this change is to technically support the user and developer expectation that app updates are, by default, only delivered from the original install source.

a. The original install source of an app can achieve this by asserting that it will be responsible for delivering all updates to an app it installs. Where an install source makes this assertion, no other install source will be able to update the app automatically, and the install source is recorded by the operating system as being the "update owner" for the app in question. It is expected that any app store would make this assertion for apps it installs.

b. Should a different install source attempt to perform an update to the app in question, it will not proceed automatically. Instead, an Android operating system dialog (i.e. a message) will be shown to the user informing them that updates are normally performed by the original install source, and giving the user the option to permit the update anyway, or cancel. Should the user decide to permit the update, the "update ownership" for the app in question is cleared, and the update proceeds.

12. Consequently, Android 14 currently allows any app store on the device to request user permission on an app-by-app basis to update an app that was installed by another app store, in cases where the original store claimed "update ownership." This request happens as part of an attempted app update, where the new store has a compatible update available, rather than in anticipation of a possible future update. The dialog through which the user permission is sought looks like this:


Audible

Update this app from Galaxy Store?

This app normally receives updates from Google Play Store. By updating from a different source, you may receive future updates from any source on your phone. App functionality may change.

13. In this example, the Audible app was originally installed by the Play Store, which claimed update ownership over the app. Because a different store—here, Samsung's Galaxy Store—is attempting to update the app, the Android operating system queries the user regarding whether to clear Play's "update ownership" over the app. The dialog box shown above is displayed by the Android operating system, not the app store, although the app store can decide an appropriate moment for it to be triggered.

14. Critically, when the "update ownership" is cleared from an app, *any app store on the device* (including but not limited to the app store that performed the update which led to the dialog, above, being shown) can then update the app. In other words, app ownership does not "change" from one store *to the other*. Instead, the user can choose to clear ownership altogether, and allow any app store to update the app in the future. One consequence of this is that, should a user choose to proceed with the update, they will not be faced with the same dialog for that same app again.

Changes to Android 14 to Implement Library Porting

15. As I understand the proposed injunction, there are two differences between the Android 14 behavior I have just described and what the library porting remedy would require.

App-by-App Restriction

16. First, the proposed injunction says that "Google shall provide Users with the ability, *subject to a one-time User permission*, to change the ownership for any or all" apps installed by the Play Store on a user's device. Proposed Injunction § II.A.1.ii (emphasis added). Android 14 does not allow update ownership to be cleared *in bulk* as the proposed injunction envisions. Instead, in Android 14, third-party app stores can request permission to clear ownership on an app-by-app basis, as and when the third-party store attempts to install a compatible update. There is no limit to the number of update requests that an app store can send a user.

17. Android 14 was designed this way to easily support cross-store app updates, while protecting users, app developers, and app stores from unauthorized or unintentional cross-store updates. By allowing users to control update ownership on a granular, app-by-app basis, in the

context of attempted updates to each app, users are presented with a straightforward decision which can be made quickly. I consider this design fit for purpose and in keeping with similar decisions that users are asked to make in the course of using their device—balancing the need for user control with ease-of-use, and not wanting to overload the user with unrelated or hypothetical decisions, which do not immediately pertain to the action they are performing. If a user wants to obtain an update to an app or game from a particular app store, having initially acquired that app or game from a different store, the Android 14 solution ensures that they can easily authorize that update. For the user to have to make a decision about other apps or games on their device at the same time risks confusion, and slows down the user from successfully obtaining the update that they had set out to install.

18. A further motivation for the app-by-app approach is that this reflects the manner in which users already make decisions about the apps and games they want to install, and from where they obtain (and update) those apps—*i.e.* in general, users make app installation decisions on an app-by-app basis, throughout the use of their device, rather than being asked to make decisions *in bulk*. Users may have legitimate reasons to prefer to have different apps updated by different app stores. For example, App Store A may offer exclusive content for one app on a user's device, while App Store B may offer promotional discounts for a different app on that user's device. Or a developer might release updates more frequently (or earlier) on App Store A than on App Store B, so a user might prefer to receive updates to the app from App Store A. Or a user might trust the policies of a particular store more than another, especially for apps they consider to contain sensitive data (e.g., health or financial apps). A user presented with a request to update all apps on the device from a single store may not realize that they have these options, and may not understand the consequences of agreeing to that request.

19. To implement the "one time User permission" requirement of Epic's proposal, which would require *bulk* ownership change, Google would have to modify the Android operating system. Specifically, Google could introduce a new Android API to request a bulk ownership change, with a corresponding "behind-the-scenes" permission that app stores would declare in their app manifest and that governs the use of this API. When this API is invoked along with a list

of app package names, the API would display a user interface (either with a dialog prompt, or a screen in the Settings app) that the app store could use to seek user consent to perform updates to one or more apps without the per-app update ownership dialog prompt. The actual change in update ownership for each app would be deferred until the third-party app store successfully installs an update for each app.

20. If ordered to make this kind of change, Google would permit the developer to choose whether it would want its apps to be subject to a bulk-ownership change protocol. In particular, Google would give developers the opportunity to indicate in the code of their APKs whether the app can be transferred in bulk along with other apps, or if instead the per-app permission would continue to apply. Google would also give the developer the opportunity to indicate, in the code of the APK itself, *which* individual third-party stores are permitted to obtain ownership over the app by means of a bulk-transfer. The consequence of this opt-in process would be that the "one-time User permission" would apply only to the apps of developers that are comfortable with this behavior, and could be requested only by the third-party app stores identified by the developer.

21. I believe this opt-in structure to be consistent with the proviso in Epic's proposed injunction that the new store would become the "update owner" of bulk-transferred apps only "if and when those apps become available on the Third Party App Store," Proposed Injunction § II.A.1.ii, which means that the developer has affirmatively chosen (i.e., "opted in") to distribute the app with the third-party store.

22. I note that Google understands the term "one-time User permission" to mean that an app store can issue a single "update ownership" request for a group of apps already installed on the phone, and not that an app store can issue a single request for permission to automatically update all apps acquired from any source in the future. The latter interpretation would lead to a host of additional problems. For example, a user that granted the future-facing permission to App Store A (quite possibly without a full understanding of the consequences) might, some weeks or months later, install a new app or game from App Store B, only to have that app instantly updated by App Store A to a different version, without any in-context consent or disclosure. This could

easily result in user confusion—for example, the user might expect to be able to spend rewards points earned through App Store B on in-app content—or in the event of problems with the app, the user might complain to App Store B, incorrectly. In the worst case, a 'tragedy of the commons' could arise in which all stores become incentivized to solicit this one-time, perpetual user permission, allowing each to freely update any app from any store, thereby eroding the user, developer, and store benefits of "update ownership," and reintroducing the very problems the Android 14 solution set out to address.

"Change" Ownership

23. The second difference between Android 14 and the language of the proposed injunction is that the injunction requires Google to implement a user permission that would "*change* the ownership" for apps downloaded from the Play Store. As explained above, the Android operating system does not allow a user to "change the ownership" of an app from one store to another, but rather only enables a user to "*clear* ownership" altogether, so that the app can be updated by *any* app store.

24. Here again, Google had good reasons for designing Android 14 this way. The Android operating system has no way to determine whether an app store actually distributes any particular app. If an app store could ask a user to "change ownership" of an app that the app store does not actually distribute, then the user could stop receiving updates for the app. This would lead to several harms.

25. First, the app developer who has no association with the third-party store would be unable to push updates out to its users, significantly harming its business.

26. Second, users would stop receiving security updates for their apps. Updates are a critical way in which developers keep their users safe from malware and other vulnerabilities. For example, in September 2022, WhatsApp announced that it had patched a security vulnerability that could have allowed attackers to remotely plant malware on a victim's smartphone during a video call. If these apps were not updatable on a user's device because the user had "changed ownership" to an app store that did not actually distribute those apps, the user's phone would have remained vulnerable to these attacks, with potentially devastating consequences.

27. None of these risks would be apparent to a user who is simply shown a dialog box asking for permission to "change ownership." The user may not even read the text of the dialog itself and merely tap through. (By contrast, Google made a conscious decision in choosing the Android 14 "clear ownership" implementation to minimize the risk should a user tap through the dialog without considering the message). Moreover, I note that these risks are even more significant when bulk transfer and "change ownership" are considered together. Android 14 addresses these risks by allowing a user to "clear ownership" but not "change ownership," because "clear ownership" means that any app store, including the original source of the app, can update the app.

28. If Google were required to implement both of these changes to Android 14, then an app store could send a one-time user permission to "change ownership" of every app on a user's device, including apps that the third party app store cannot update. At that point, every app on a user's phone will be incapable of updating, including apps that are integral to the functioning of the phone. Because, as noted, security patches are a common reason for pushing out an update of an app, this scenario dramatically increases the security risks on the user's phone. Moreover, these changes together not only create risks to users who unknowingly approve future updates by a non-preferred store, but also would create an opportunity for hostile store developers (or opportunistic developers of malicious apps that are not in fact stores) to actively block updates. These are precisely the risks that we try to guard against when building Android APIs.

29. These risks are further magnified when library porting (the two changes described above) is considered in tandem with the distribution of third party app stores remedy discussed below, in the absence of any of the implementation safeguards discussed in this declaration. If both of these remedies were implemented as phrased in the injunction, any Android user visiting the Play Store could be two taps away from inadvertently shutting off updates to all apps on their phone—one tap to install a third party app store from Play, and one tap to provide a one-time user permission to transfer ownership of all apps on the phone to the third party store, even if the third party store cannot update the apps.

30. Despite these risks, if Google were ordered to make the change envisioned by Epic's proposed injunction, this is my current assessment of how Google would do it. First, Google would create a new code path in the operating system to perform the update owner switch, subject to a documented API behavior change for apps using the PackageInstaller API (this may also include a new API to allow updaters to indicate whether they want to claim ownership when updating apps which currently have no update owner). Second, Google would create a new update ownership dialog, including a new design and language to accommodate this change in behavior (both for the case of an installer wanting to perform an update and switch ownership, and the case of an installer wanting to perform an update but not switch ownership, *e.g.*, a browser-downloaded APK update).

31. Finally, to address the risks described above regarding change of ownership, if Google were ordered to implement this remedy, then Google would also implement a similar developer opt-in protocol described above. Specifically, developers could embed a statement inside the APK file indicating whether ownership of the APK may be transferred, and if so to which particular app stores a change of ownership may be accomplished. OEMs and carriers could configure the same opt-in permission for apps that they preload.

32. As with the bulk-transfer permissions discussed above, I believe this opt-in structure to be consistent with the proviso in Epic's proposed injunction that ownership must change the new store only "if and when those apps become available on the Third Party App Store," Proposed Injunction § II.A.1.ii, which means that the developer has affirmatively chosen (i.e., "opted in") to distribute the app with the third-party store.

Signing Keys

33. I note that developers must take certain actions with respect to their apps to allow for the possibility of third-party app stores updating their apps. This point applies to existing Android 14 and future versions of Android reflecting any of the above changes.

34. Android requires that all APKs (i.e. app packages) be digitally signed with a certificate before they are installed on a device or updated. When an installer attempts to update an app, Android will compare the certificate of the existing, already installed app to the certificate

of the updated version of the app to ensure that the certificates match. If the certificates match, the update can proceed. If the certificates do not match, the update is blocked.

35. App signing is a critical security protection that has been in place since the beginning of Android. It not only gives users (and in some cases the developers of other apps) the assurance that the app is in fact from the intended developer, but also prevents hostile actors from updating apps with a malicious version of the same app. In the absence of app signing, an installer could "update" App A from Developer X to App B from Developer Y. App signing reduces that risk by ensuring that the updated app bears the same signature as the app-to-be-updated.

36. Because Android uses app signing to authorize updates, the loss or compromise of a developer's signing key has severe consequences. If a developer no longer has access to a signing key, they may be unable to release updates to apps that are already installed on user devices. If a bad actor gains access to a developer's signing key, the bad actor may be able to distribute a malicious update to the developer's app.

37. For many years, developers that distributed on Google Play were responsible for managing the signing keys for their own apps, securely holding these keys, and uploading signed APKs to the Play Console for distribution. Since 2017, with the introduction of "Play app signing," developers have been able to let Google Play securely manage and protect the signing keys for their apps, and sign their APKs for distribution. In addition to the security benefits, Play app signing allows Play to generate and serve optimized APKs for each device configuration, which helps to make APK downloads as small and fast as possible. Developers of new apps can choose to use a Google-generated signing key, or upload their own key for use. Most developers opt for a Google-generated signing key so they do not have to worry about key loss or compromise, instead relying on the security of Google's key management service. For that reason, when a developer elects to have Google generate and safeguard the signing key, Google does not provide that signing key to the developer. This avoids the risk of key compromise, meaning that a bad actor cannot obtain the signing key from Google or from the developer. Since August 2021, most new apps have been required to use Play app signing for the same reasons that

developers often chose to do so before—Google can safeguard the key and protect it from loss or compromise, and developers can benefit from optimized APK distribution.

38. If a developer chooses to have Google sign the app using a Google-generated signing key, and then the developer releases the app on another store with a different key (e.g. a key generated by that store or the developer), then Android will not recognize that the two apps are, in fact, the same app. Instead, because the Play version of the app and the third-party-store version of the app have different certificates, they will be considered two different apps. Accordingly, the third-party store will not be able to update the Google Play installed app (or vice versa). To address this issue and allow cross-store updates, the developer will need to ensure that the Google Play version of the app and the third-party store version of the app are signed with the same key. The developer can accomplish this by opting to upload their own key for use with Play app signing—either when first publishing their new app on Play, or, in the event that the developer had previously opted for a Google-generated signing key for their app, by using the Google Play Console to perform a key upgrade (also known as a key rotation).

39. If a developer originally chose to have Google sign the app they distribute on the Play Store with a Google-generated signing key, and then the developer distributes a different version of their app to a third-party store with, for example, a different billing system, Android will not allow the third-party store to update the Play-signed app (and vice versa), because they are not recognized as being the same app. Instead, the user would have to uninstall the app, and re-install it from the third-party store. In this circumstance, as I describe above, Google would allow the developer to use the Google Play Console to "upgrade" their Play signing key to a new key that the developer creates and uploads (and thus retains a copy of). That will allow the developer to use the same signing key when they distribute their app through a different app store, which will allow the third-party app store to update the app.

Costs and Time to Implement

Costs

40. Based on my experience working with a team to implement changes to Android on numerous occasions, I estimate that these changes to the Android operating system would require the following resources.

Resources / FTEs	**Duration Required**
Solution Build	
2 software engineers	1 year
1 user experience designer	3 months
1 user experience researcher	2 months
1 product manager	6 months
1 developer relations engineer	1 month
1 technical solutions consultant	3 months
Ongoing Maintenance & Support	
1 software engineer	2 mos./yr (2-6 years)

41. For building the solution, my estimates above are based on the following tasks.

42. The 2 software engineers would design and implement the developer consent schema, as well as the Android operating system parsing for this schema, for both ownership transfers and bulk ownership change, accounting for the complexities of store identity and ease of implementation on the part of the app developer. They would also design and implement the bulk ownership change permission and associated APIs and user interfaces; design and implement the ownership change mechanism and associated APIs and revised user interface, as well as OEM configuration for preloads; and implement testing for all of the above, in particular with consideration for compatibility of apps that expect the Android 14 behavior; and work on technical documentation for developers.

43. The 1 user experience designer would design the new user interfaces and permission flows. The 1 user experience researcher would run a study to test user comprehension

of the new user interfaces and flows. The 1 product manager would drive all of this work, in particular with a regard for compatibility and security. The 1 developer relations engineer would work with the engineers on technical guidance for app and store developers, produce sample code, and engage with developers who provide feedback or encounter issues. And the 1 technical solutions consultant would provide documentation to OEMs, engage with their feedback, and identify edge-cases relating to preloaded apps or stores which may influence aspects of the technical implementation.

44. For ongoing supervision and maintenance, the equivalent time of 1 software engineer would maintain tests and test infrastructure; keep the new implementation compatible with future, and potentially unrelated, Android operating system changes; and triage and fix bugs that are identified either through automated testing or external feedback from app or store developers.

45. These estimates are influenced by my experience in developing the Android 14 "update ownership" feature, which involved changes to similar aspects of the Android operating system (including API behavior changes and new user interfaces), with a comparable scope to the changes proposed for this remedy.

Time to Implement

46. The changes required for this remedy would take a substantial amount of time to implement. In this case, I estimate these changes will take roughly one year to complete.

47. In general, changes to the Android operating system are enormously consequential. The operating system is the underlying software that powers Android phones. An error or bug in the operating system can have disastrous consequences for users, developers, OEMs and Google. Accordingly, changes to the Android operating system require extensive research and due diligence, internal testing, developer previews, beta testing, feedback from users and OEMs, and bug fixes prior to a public release—and on an ongoing basis thereafter. Google performs these tasks and releases new major versions of Android on an approximately annual cycle.

48. Although Google performs more minor changes to Android on a more frequent cadence, features that involve behavior changes to APIs and impact external app developers—such as those at issue in this declaration—are saved for our annual release cycle because of the increased risk to system stability, performance, and app compatibility, and the greater burden that is placed on app developers to adjust for API behavior changes. Some examples of minor changes that are made on a more frequent cadence include, in a recent minor release: we enhanced screen sharing functionality by allowing a user to select a single app window that they wish to share, rather than their entire screen; we added the option from the device's "quick settings" panel to share Wi-Fi credentials; and we updated the user interface of the Settings app to show the "package name" of installed apps in order to aid diagnostics. It is also worth noting that changes that Google makes in minor releases to the Android operating system are much less commonly adopted by OEMs.

49. Making these changes off cycle immediately, or as part of our more frequent Android updates for less significant changes, could create serious user-experience and security issues. New features implemented in the Android operating system itself, like those addressed in this declaration, take time to build in a way that avoids unintentional regressions in device functionality, including unforeseen interference with the operation of users' apps. For example, in previous developer previews and beta tests we have done, there have been multiple publicly reported instances of malfunctions in the sideloading flow, such as where the "install unknown apps" permission does not persist for a given installer, or where enabling that permission has crashed the app. It takes time to test out the changes and establish the possible app compatibility impact, and also time for impacted app developers to either make necessary adjustments to their apps, or report issues to Google. When it comes to automated testing, it often takes as much time to write the test code as it does to implement the Android feature itself.

50. An important component of our testing program involves a public developer preview and beta programs to ensure that the changes operate as intended, which usually occurs over the course of a few months. We solicit feedback from these users and developers, which helps us identify bugs and other issues. There is no way to fast track that process because it

requires a period of time where users are using their devices in the ordinary course, as part of their daily lives, and reporting bugs or issues, and developers are testing their apps out, considering the impact of API behavior changes, and exercising new APIs, and similarly reporting any issues to Google.

51. The developer preview and beta process also serves an important security purpose. Any new operating system feature or API change invariably carries a risk of newly added vulnerabilities, either resulting from implementation bugs or unanticipated design flaws. While Google goes to great lengths to prevent such vulnerabilities being introduced in the first place, the public testing period before the final public release of a new Android version provides an opportunity for external security researchers to report vulnerabilities to Google, and for Google to fix these vulnerabilities before they risk exploitation on end-user devices. Furthermore, Android's vulnerability reward program (which facilitates the responsible disclosure of security or privacy vulnerabilities, and pays reporters of these issues, depending upon the severity) has sometimes offered bonus payments for vulnerabilities found in beta releases, in order to incentivize the discovery of such issues.

52. Another important aspect of the testing process involves OEM testing. OEMs are the ones ultimately choosing whether and how to adopt Android changes in their updates or new releases. While I am not directly involved in this process, I understand generally that this involves OEMs engaging in significant engineering work to assess and integrate changes, possibly adapting them for their own needs, and then implementing their own testing program, often running beta programs of their own to get feedback from early adopters. In many countries, there is even another testing step involving the mobile carriers, many of which undergo their own stage of technical acceptance testing. This aspect of the testing process involving OEMs and mobile carriers is out of Google's control, and contributes to the lengthy timeline for rolling out Android changes.

53. My resource requirement estimates above assume that the Android changes discussed in this declaration would occur as part of Android's normal annual update cycle. If Google were ordered to implement these changes off-cycle, the cost to Google would be far

higher, as Google would have to initiate a separate round of user, developer, and OEM testing and feedback described above. Moreover, performing these changes to Android off-cycle would require us to modify the implementation approach in ways that risk compromising operating system stability, app compatibility, ease of adoption by developers, and create uncertainty as to whether the changes would actually work correctly in all circumstances.

<u>Distribution of Third Party Stores Through Play</u>

54. In this section, I describe the changes to the Android 14 operating system that would be necessary to implement Epic's proposed remedy relating to distribution of third party app stores through the Play Store.

<u>Required Changes to Android</u>

55. The proposed injunction requires that the download process for third-party app stores distributed through Play be the same as the process for any other Play-distributed app, except that Google may present the user with a "single one-tap screen asking the User to allow the Third-Party App Store to install other apps." Proposed Injunction § II.A.2.i.

56. To implement this functionality would require changing the Android operating system. This is because Android currently has a different install flow for app stores that are pre-installed as compared to app stores that are subsequently installed by the user.

57. As noted above, Android has two permissions that allow an app to install other apps. The first permission is called the INSTALL_PACKAGES permission. This permission is granted to pre-installed app installers by the OEM at the time the Android device is configured by the OEM. For example, on Samsung Galaxy devices, both the Samsung Galaxy Store and the Google Play Store have the INSTALL_PACKAGES permission. The second permission is called the REQUEST_INSTALL_PACKAGES permission, which (assuming sideloading is supported on that device) can be used by any app if the user consents to granting that app installer rights. On Android, any app can technically request the REQUEST_INSTALL_PACKAGES permission, because the operating system itself has no concept of appropriate types of app that can act as an install source. For example, an app that displays the weather could in principle ask the user to give that app REQUEST_INSTALL_PACKAGES, should the developer configure this. As I

describe later in this declaration, at paragraph 76, "hostile downloaders" commonly take advantage of this fact to trick users into installing malware.

58. When an app that has the REQUEST_INSTALL_PACKAGES permission (e.g., a sideloaded app store or other app that the user has enabled to install other apps) attempts to install an app, the user will receive a confirmation dialog ("Do you want to install this app?") each time the user attempts to install an app. When an app that has the INSTALL_PACKAGES permission attempts to install another app, the confirmation dialog is not shown.

59. If an app store is distributed through Play, then it will be able to receive the REQUEST_INSTALL_PACKAGES permission (not the INSTALL_PACKAGES permission) because, by definition, that app store will not be a pre-installed store and will not have been configured by the OEM with the INSTALL_PACKAGES permission. Accordingly, the Play-distributed app store would receive the confirmation dialog ("Do you want to install this app?") each time the user attempts to install an app.

60. Because Epic's proposed injunction requires that Google provide Play-distributed app stores with the same install experience available on Play, Google would need to remove this confirmation dialog.

61. To remove this confirmation dialog, Google would likely extend a change that was made in Android 12, which removed the need for user confirmation for app *updates* (under certain circumstances), to apply also to *first-time installs* of apps. Google would couple this change with a new behind-the-scenes permission (*i.e.*, not something the user is made aware of explicitly or has to grant themselves) that app stores would have to add to their manifest, similar to the permission that was added for the Android 12 change relating to app updates. This would be in addition to the user-visible REQUEST_INSTALL_PACKAGES permission.

62. However, simply removing the confirmation dialog, without more, would create security vulnerabilities for users. This is because sideloaded installers could use this capability to silently install harmful apps that jeopardize the user's security, without the user's knowledge. For example, if the user has ever consented to allowing App A to install other apps (i.e. granted the

REQUEST_INSTALL_PACKAGES permission), App A will in the future be able to install additional apps without any consent by or even notification to the user.

63. To mitigate that risk, Google would likely add a technical restriction that the new permission would be granted, behind-the-scenes, by the installer of the app store itself. That would mean that if Play is the source of the third-party app store (as envisioned by Epic's proposed injunction), Play would grant the third-party app store the privilege that exempts that app store from the per-app confirmation dialogs.

64. In addition, to further lessen the risk of silent installs of harmful and unwanted apps in the background, Google may also require that installing a new app (without a confirmation dialog) be permitted only in response to a proactive install decision taken by the user (e.g. tapping an 'Install' button that the store renders in their user interface). The intention of this requirement is to match the user expectation that new apps are only installed from a third-party store when the user takes an action to initiate this from within the store app.

65. Finally, to show the "single one-tap screen asking the User to allow the Third-Party App Store to install other apps" described in Epic's proposed injunction, Google would also change the Android code, and would create this new screen as part of the Google-signed PackageInstaller app.

Costs and Timeline to Implement

66. To implement the above described changes to Android, I estimate it would require the following resources.

Resources / FTEs	**Duration Required**
Initial Build and Implementation	
1 software engineer	1 year
1 user experience designer	3 months
1 user experience researcher	1 month
Ongoing Maintenance	
1 software engineer	1 month per year

67. For the initial implementation, my estimates are based on the following tasks. The 1 software engineer would likely design and implement the new manifest permission and associated API for Play to grant the permission; design and implement the "one-tap" screen flow for granting the REQUEST_INSTALL_PACKAGES permission, while accounting for OEM customizations that currently exist in their Settings app implementations; research, design and implement the mechanism to ensure that installs without OS enforced user confirmation occur with some proof of user interaction with the third-party app store; implement automated testing for all of the above; and produce technical documentation for developers. The 1 user experience designer would design and iterate on proposals for the new one-tap screen. And the 1 user experience researcher would conduct a user study considering the comprehension of the new one-tap screen, as well as users' reactions to installs from third-party app stores without OS-enforced confirmation.

68. For ongoing maintenance, the 1 software engineer would maintain the tests and test infrastructure; keep the new implementation compatible with future, potentially unrelated, Android operating system changes; and triage and fix bugs that are identified either through automated testing or external feedback from app or store developers.

69. I estimate that the time it will take to implement these changes in Android is roughly one year. As explained above, at paragraphs 47-52, any change to the underlying operating system is enormous consequential, and takes significant time to finalize.

Third-Party App Stores Generally

70. In this section, I offer some observations about third-party Android app stores based on my experience working on Android security. During my time at Google, I have spent time observing certain trends and behavior by third-party stores.

71. Some public reporting indicates that, as of 5 years ago, there were over 400 Android app stores that were generally available. *See* Forbes, *The 'Other' Android App Stores - A New Frontier for App Discovery*, https://tinyurl.com/3kws69sr. Another source reported over 300 Android stores. *See*, Business of Apps, *App Stores List*, https://tinyurl.com/mryk9rwb. I am

aware of dozens of these app stores myself. Some of these third-party app stores appear to be vectors for an elevated volume of pirated apps, malware, or inappropriate content.

72. A significant example of pirated apps is HappyMod, one of the most downloaded independent third-party app stores. HappyMod is dedicated to distributing "modified" versions of Android apps and games, many of which are merely apps that have been "scraped" or "repackaged" to allow users to obtain a paid app for free or to "unlock" premium in-app content items for free without paying the developer (e.g., a modified version of Netflix that allows you to get content for free without a subscription). For example, HappyMod distributes modified "free" versions of Minecraft and Grand Theft Auto San Andreas, while those apps are available on Play for $6.99 and $6.99, respectively. This is supported by the feedback we hear from top game developers, who express dissatisfaction with the scale of unauthorized distribution on Android, in contrast to iOS.

73. Like any app store, a third-party store may be a vector for malware, and many such stores do not have the same security processes as Play does. The Aptoide app store, for example, is widely known to distribute harmful apps. APKSOS.com is another example. In one publicly reported example in November 2022, it appears that a group of malicious apps that were designed to initiate unauthorized financial transfers from a user's device were removed from the Play Store, but APKSOS.com continued to host one of those apps in its app store. SOCRadar, *Third-Party App Stores, Risks and Precautions*, https://tinyurl.com/25djye76. That app, "Phone AID, Cleaner, Booster 2.9 APK," in fact is *still* available on APKSOS.com. In other cases, third-party store developers themselves have malicious intentions. One example was CepKutusu.com, a Turkish app store, which was intentionally designed to spread banking malware with every app downloaded from its platform. In the course of my work I have also seen an "app store" operated by a state-sponsored hacking group, with the sole purpose of delivering spyware to targeted individuals that pretends to be a legitimate messaging app.

74. As for mature content, one such example is the Nutaku Android store, which advertises itself as "the world's largest 18+ gaming platform," and features apps with adult

content. Aptoide also carries adult apps including Pornhub and an unrestricted version of Telegram that allows adult content.

75. I cannot say how prevalent these issues are across the hundreds of third-party app stores, but there are at least some that distribute pirated, unsafe, or inappropriate content like the above examples.

76. Beyond third-party app stores specifically, I have been aware of many thousands of apps whose primary purpose, unbeknownst to the user, is to act as a distributor for other malicious apps. These types of apps are known as "hostile downloaders," or "droppers," and while they typically do not claim to be app stores—commonly they present themselves as utility apps, such as file managers or QR code scanners—from a technical perspective they are indistinguishable to app stores, as far as the Android operating system is concerned. Hostile downloaders remain a present-day threat for Android users that Google works hard to defend against through improvements to the Android operating system, as well as our malware detection systems. A recent example of a hostile downloader campaign that has sometimes bypassed Google's defenses is the "Anatsa" banking trojan, which is first installed by a hostile downloader, and subsequently attempts to exfiltrate banking credentials in order to steal money from users. Should Google be ordered to make third-party stores available on Play, and enable the new install flows for these stores, it is essential that Play has policies in place to determine which apps are legitimate third-party app stores and are consequently eligible to take advantage of these new capabilities.

77. Finally, I note that these observations above reflect some of my understanding of the risk landscape of third-party stores in the present day. If Google is ordered to implement the remedies discussed in this declaration, including a separate remedy requiring Google to give its app catalog to third-party stores, that could change the landscape significantly by augmenting the volume of these potentially problematic third-party stores and empowering them with new capabilities that put users at risk.

* * *

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

Executed on this 24th day of June 2024 (Pacific Time), in London, United Kingdom.

DocuSigned by:
Edward Cunningham
A3D1EA4B9E7D484...

Edward Cunnir