Open
Web Advocacy

Digital Markets Act - Interventions - Web App Install on Android

VERSION 1.0

Open Web Advocacy
contactus@open-web-advocacy.org

1. Table of Contents

2. Introduction

In 2015, Google shipped the first version of Web App Default Install Prompts. The initial implementation was limited and did not integrate well with Android. A follow-up implementation in 2017 fixed these flaws, allowing for deeper system integration and improved installation success rates. This system, known as "WebAPKs", relies on Google's infrastructure and Android system components and since launch has been restricted for use by Chrome, despite requests by other vendors including Mozilla for equal access. This caps the potential of Web Applications on Android and prevents competing browsers from benefiting from the engagement that comes from Web Applications, harming their chances in the market.

We believe that under Recital 43, Recital 53, Article 6(4), Article 6(6), Article 6(7) & Article 13(4) Google is obligated to provide this functionality to third-party browsers.

To install richer Web Apps, Chrome requests a package from a Google-run server using a Play Services API that is restricted to Chrome. The server then sends an Android Package (APK) for the Web App and Play Services installs it. This process is known as "WebAPK Minting", but since no other browser can access the minting API, the ability to properly install Web Apps is kept exclusive to Chrome (except Samsung Browser on Samsung devices).

WebAPKs remaining exclusive to Chrome is anti-competitive, restricts browser competition and damages the viability of Web Apps because:

1. Other browsers can not compete to provide better functionality for Web Apps, for enabling better implementations of backup and restore (see below) or introducing Web App Stores.

2. Provides an unfair advantage to Chrome on Android through increased engagement from Web Apps.

3. Damages adoption of Web Apps by making them not work properly in other browsers.

Google publicly stated almost 7 years ago that they were working on sharing this functionality with other browsers but there has been no progress. We believe that they should be forced to share this functionality, at a minimum, with "Trusted Browsers". "Trusted Browsers" is our proposal for a new security standard for browsers which will set a minimum bar for security in return to gaining access to sensitive functionality on operating systems.

Should Google decide that they can mitigate the security risks associated with WebAPK minting we would encourage them to share the functionality with all browsers on Android.

It is essential that Google allows competing browsers to properly install Web Apps. The simple, timely solution to this is for Google to make the existing Play Services API available to (at a minimum) trusted third-party browsers, allowing them to use the same WebAPK minting service that Chrome enjoys access to.

There is a very clear argument that under the DMA that Google is obligated to share this API with third party browser vendors.

2.1. Backup / Restore / Transfer / Sync of Web Apps

In addition to restrictions on installation, the fidelity of Web App implementation on Android favors Google's native application ecosystem, even for browsers that enjoy access to WebAPKs today. One key gap is the failure of Android backup and restore systems to transfer Web Apps to new devices, or sync them.

That is, when users transfer their data to a new phone, all installed Web Apps are lost. This deals a significant blow against Web App viability on Android and provides reasons for business users to avoid them. Firms invest huge amounts in app installation, and the prospect of erasure during a system restore is a serious issue for them, as it results in meaningfully lower engagement and/or increased traffic acquisition costs. So long as the playing field is uneven, business users are unlikely to use Web Apps as their primary distribution method.

This undermines Web Apps viability on Android, reducing business user value when investing in platforms that compete with Google's proprietary Android and Play platforms. It's difficult to say why this situation exists, but we note that Native Apps are discovered through, and share revenue with, the Google Play store – a designated CPS of Google.

There are three possible methods that could be used to backup / restore / transfer and sync Web Apps between devices. They are:

1. **Using Google Play Services App Backup and Restore**
   All installed Web Apps, regardless of which browser installed it, need to be included as part of this backup and restore procedure as well as the newer sync functionality that was recently introduced.

   It is our understanding that this would be a minor change for Google to introduce the backup/restore as Google Play is the service that creates the WebAPKs.

2. **Using a Browser**
   A user logs into an account with a browser on multiple devices and the browser can then use APIs to manage web apps to sync them across multiple devices.

3. **Through third party backup and restore solutions**
   If and when google implements support for third party backup and restore solutions for Android, Web Apps must be included in this functionality

For Web Apps to compete fairly against the Native Apps of Google's Play Store (also a designated CPS), this must be fixed. We outline in this section why Google should be obligated under the DMA to fix this.

3. What is WebAPK Minting?

A WebAPK is a thin wrapper Native App that provides a splash screen, system launcher integration, and system settings configuration points. When launched, a WebAPK essentially starts a tab in the browser which installed the WebAPK, loading the specific URL the Web App developer configures.

All Native Apps on Android are distributed as APKs, either via the Google Play Store or via sideloading. WebAPKs allow Web Apps to be integrated into the OS for the purposes of discoverability, permissions management, shortcut creation, and uninstall.

Android's security model is built around signed native APKs. In order for Web Apps to integrate properly on Android without significant architectural changes, Web Apps need to be wrapped in a signed native APK. This allowed Google to support Web Apps across existing versions of Android without having to introduce a new architecture to support Web Apps and wait for years for it to be updated.

This strategy allows the operating system to treat Web Apps as Native Apps because, as far as the operating system is concerned, they are. WebAPKs minted by Google's servers are signed by the same keys as those of Native Apps distributed through the Play store. These keys are pre-trusted by the device as a condition of Google's licensing of Play. This also ensures that no side-loading warning messages are displayed.

It is worth noting, one large difference between the iOS and Android ecosystems is the relative prevalence of OS updates. While recent versions of Android from certain vendors have received slightly longer security patch periods, most Android users do not receive new versions of Android within a year of their release, and most Android devices only ever receive a single OS upgrade.

In essence, Google cannot upgrade Android devices that have already been sold, by adding new features to the next version of the OS. Google must rely on other mechanisms – like Play Services, also known as "GMS Core", to introduce new capabilities to the 2 billion+ Android devices in service around the world.

This is important as due to significant issues with Androids distribution via OEMs (original equipment manufacturers) resulting in the inability to patch Android, Google has been using GMS Core to deliver Android OS APIs to devices that would otherwise be un-updatable.

3.1. What are Digital Signatures?

The objective of digital signatures is to authenticate and verify documents and data. This is necessary to avoid tampering and digital modification or forgery during the transmission of official documents.

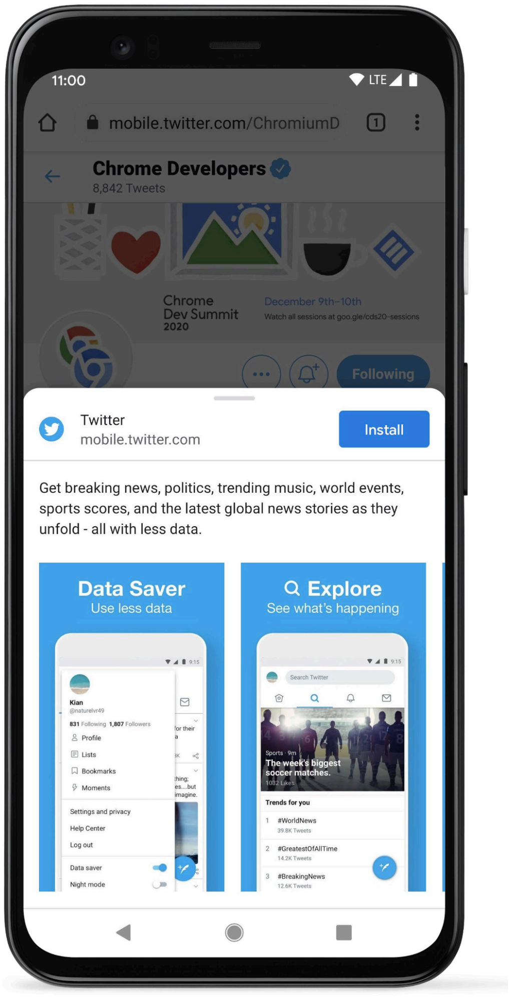
This is relevant as digital signatures are how apps on Android are authenticated and the core purpose of WebAPK minting is to allow Web Apps to use Android's existing authentication system for Native Apps.

This requires the signer to have private keys to sign apps, and the devices to have a corresponding public key (pre-trusted) to check they trust apps. The private key is kept secret by the developer (in this case Google), the public key is preinstalled on the Android phones and there is no security risk if third parties gain access to view the public key.

Essentially the private key can create signatures and the public key can check they are valid. It is by this method that the Native wrapper apps for Web Apps, are signed by the WebAPK minting server which has Google's private keys, and thus trusted by Android.

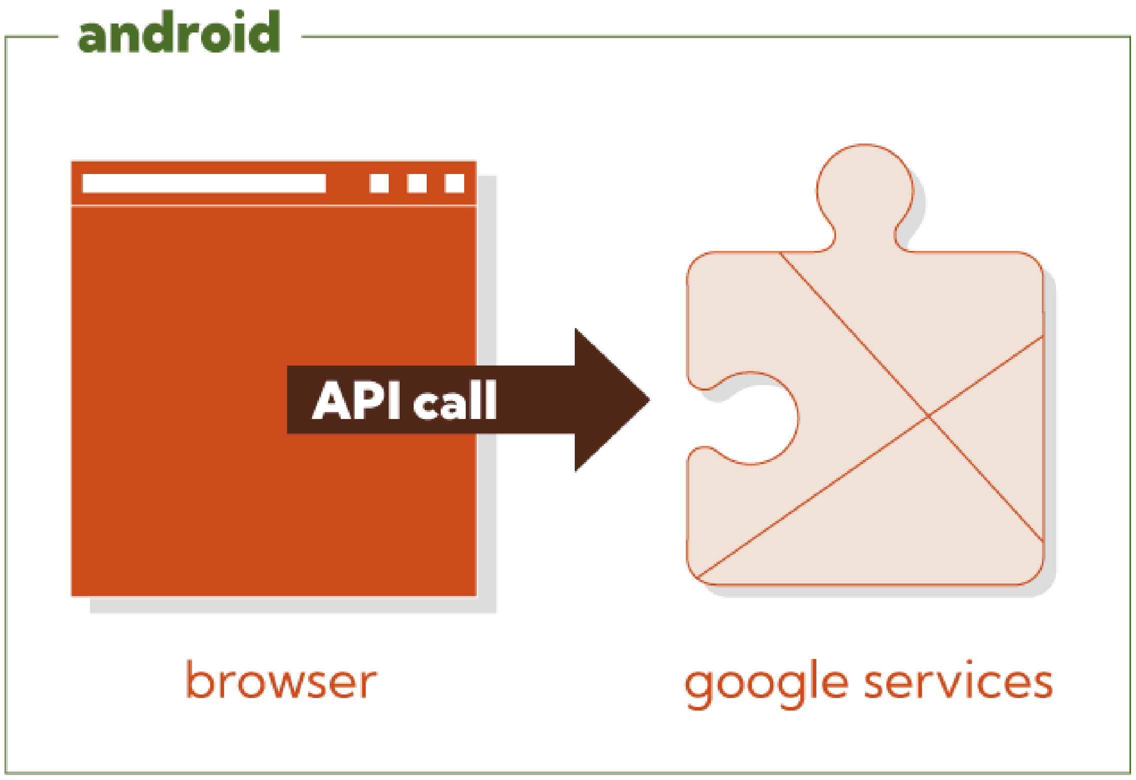
3.2. How does WebAPK Minting work?

1. The user visits a website that can be installed as a Web App. The user clicks install on an install prompt in their browser, in this case Chrome.

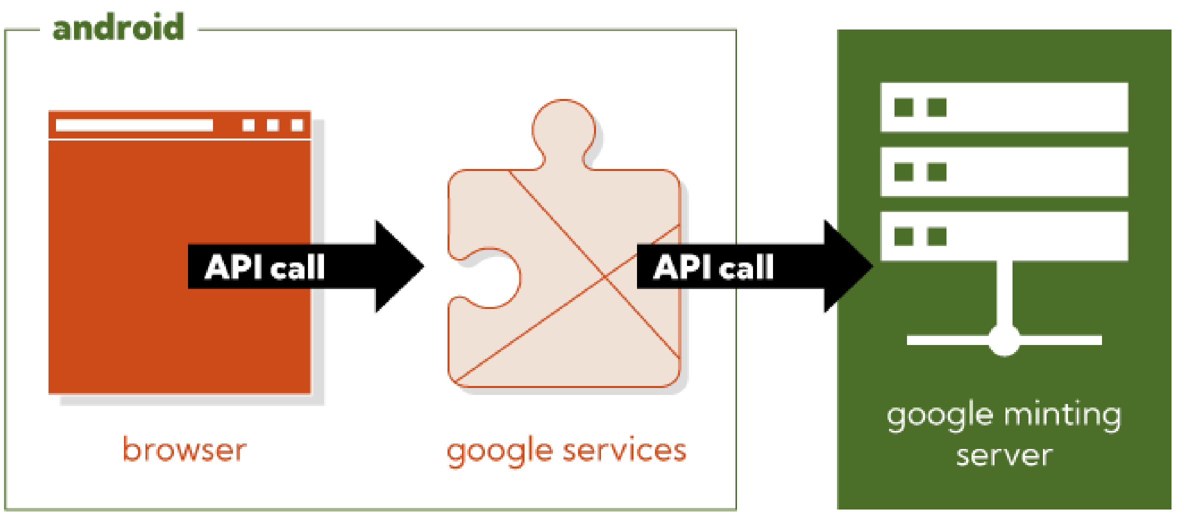


Get breaking news, politics, trending music, world events, sports scores, and the latest global news stories as they unfold - all with less data.

2. The browser calls a **Google Play Service API** on the phone. This API can identify which Browser is calling it and whether it has permission to call it. Currently only Chrome has permission



3. Google Service sends a request to a Google Minting Server

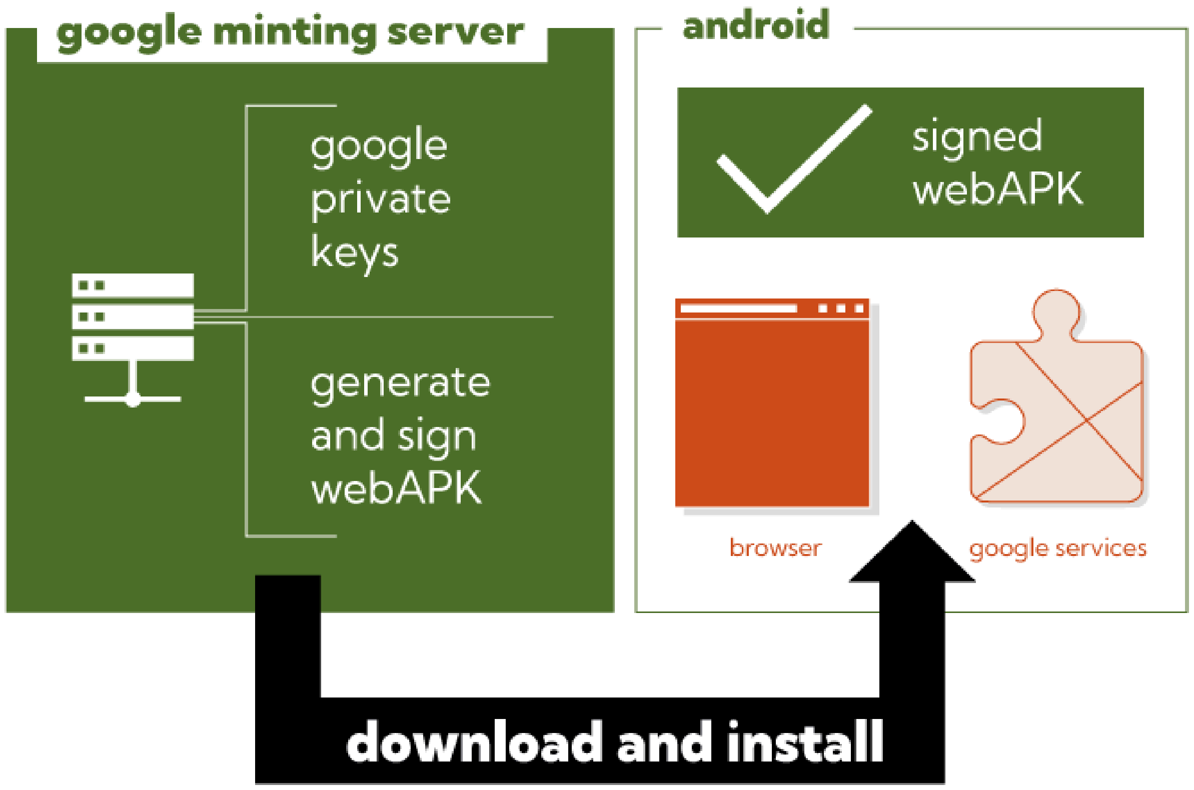


4. The Google Server generates the WebAPK with that browser and url.

Essentially this is an wrapper Android Native App with a small meta-file containing a browser identifier, the url, the icon and some other details.

5. Using the same private keys that Google signs certain Google Play Store Native Apps, pre-trusted by the device, the Google Server signs the Wrapper WebAPK App

6. This APK is downloaded and installed by the phone. Importantly it is trusted as it is signed in the same manner all other native Google apps are. The signature is checked against a corresponding google public key that is preinstalled on the device.



7. When the user opens the app, it loads and calls the listed browser to run and manage the specific URL in the manifest. Chrome also checks the signature of the signed WebAPK against one of Google's public keys to make sure it was signed by Google Minting Server.

4. Advantages of WebAPK Minting

Alternative Android browsers cannot install Web Apps using WebAPK minting today, but they can add “bookmarks” to the home screen. This bookmark system was the original implementation that Chrome launched in 2015. The deficiencies with this system were so large that WebAPKs were invented.

WebAPKs are improvements over “shortcut apps” in many ways. They:

- Can be found in the Android app drawer
- Are listed in the 'All apps' section of Android settings
- Provide their own 'App info' pane, just as native apps do
- Report resource usage in Settings, including:
    - Storage
    - Data usage
    - Battery consumption
- Can be updated by developers after installation, including app icon and name, note the user is notified via a prompt for security reasons
- Receives its own target URL space for its domain, so when a user clicks on a link within the scope of the Web App, it will open the installed web app instead of the browser.

Note, not all resource use (Storage, Data usage, Battery consumption) is correctly reported for Web Apps installed via WebAPKs today. We understand this is because Android has failed to provide sufficient APIs for browsers to provide such information to the OS for attribution purposes. For Web Apps to achieve equality, we believe this must be rectified by the Android team, but do not believe that it should not be allowed to delay opening of access to the WebAPK Minting process to competing browsers.

5. Google has not shared WebAPK Minting

On Android, only Chrome has the ability to mint (create) WebAPKs through its exclusive access to the required Google Play Store APIs. All other browsers must create a bookmark. The exception is Samsung, which has created its own minting service for their Samsung Internet browser via the Galaxy Store. This is covered here.

At the bottom of this article (written by Google) it mentions the following:

> ***"I am a developer of another browser on Android, can I have this seamless install process?***
>
> *We are working on it. We are committed to making this available to all browsers on Android and we will have more details soon."*

This article was written May 21st 2017 **(more than 6 years ago)** but there has been no update from Google.

We are aware that multiple browser vendors requested WebAPK Minting but were rebuffed, including Samsung, Opera, Mozilla, Edge, Brave, and Kiwi Browser.

A more recent bug in the Chromium bug tracker, likewise, has remained open for multiple years without commitment from Google despite multiple browser vendors (Brave, Microsoft, Kiwi) commenting that they would benefit from access:
https://crbug.com/1243583

6. Why is this a problem?

This issue is important for a number of reasons:

- Damages Competition between Browsers
- Limits Choice for End Users
- Damages the Web App ecosystem
- Is against Open Web Standards

6.1. Damages Competition between Browsers

By controlling WebAPK minting, Google stifles innovation in the browser market. Other browsers cannot offer the same seamless and native-like PWA install experience as Chrome on Android, limiting their ability to compete. This behavior unfairly allows Google to provide better functionality for Chrome than other browsers on Android phones. It also means that only Google can decide what functionality Web Apps should have and makes it impossible for other browsers to effectively compete in the provision of Web App features.

6.2. Limits Choice for End Users

The limits users choice of browser when it comes to Web Apps. They are essentially forced to use Chrome if they want the best Web App experience. Users should be free to choose browsers that offer better Web App functionality but this is impossible if other browsers are prevented from installing them.

6.3. Damages the Web App ecosystem

Web Apps are the only open and interoperable competitor to the Android and iOS App Stores. Allowing them to flourish to apply significant competitive pressure on the mobile App Stores.

Opening up WebAPK minting would benefit developers and businesses by making it easier for them to reach a wider audience with their Web Apps. This can result in cheaper, higher quality software that automatically works on all major platforms.

Web Apps installed through other browsers are not as integrated with the Android system as those installed through Chrome. This can lead to a less user-friendly experience for consumers. Some features of Web Apps do not work as well or at all when installed through other browsers. This can limit the functionality and usefulness of Web Apps for consumers.

It is worth noting that according to Mozilla's study a well designed choice screen (such as required by the DMA) would drop Chrome's market share on Android to between 54-62%, , down from nearly 90%, significantly decreasing Chrome market share on Android. This will greatly increase the importance of third party browsers being able to support Web Apps.

6.4. Against Open Web Standards

One of Google's stated goals is to promote open web standards, and they are a supporter of the open web foundation, yet they contradict this by keeping WebAPK minting closed and exclusive to Chrome.

Google has claimed that they are working on opening up WebAPK minting to other browsers. However, there is no clear timeline for this, and they have not provided any specific details about their plans. In our view, given it has been more than 6 years since this statement, without regulatory pressure Google will never share this functionality with third party browsers.

It is important that this clear violation of the DMA is rectified by Google's March 6th 2024 deadline. We outline why in this section. This will ensure that all browsers on Android can offer a seamless and native-like PWA experience, and that consumers have the choice and freedom to use the browser they prefer. Further it will allow Web Apps to work well on all browsers that choose to implement them, not just Chrome.

7. How Can Samsung Implement WebAPKs?

We have come to learn that Samsung Internet requested access to the WebAPK service and, upon failure by Google to provide it, reverse-engineered the protocol for the Samsung Internet browser:

This is possible for Samsung Internet on Samsung devices for the same reason that Samsung can place the Galaxy Store on Samsung devices. OEMs may place additional “trusted” keys into the devices they sell alongside the keys that Google mandates all Google Play conforming Android devices embed. These keys are what allow “superpowered” programs like Google Play Services (“GMS Core”) or the Galaxy Store to install other Android programs.

Competing browsers (Firefox, Opera, Edge, etc.) do not have the ability to change system software, and even Samsung Internet cannot install high-quality Web Applications outside of Samsung devices. The only entity with the scope and infrastructure to facilitate a fair playing field is Google, which has repeatedly declined to do so.

While Samsung is not covered as a gatekeeper under the DMA it should be noted that Samsung had the opportunity to fix this anti-competitive issue on their devices (about 24% of smartphones) by sharing this access with third party browsers, and chose not to. That said, given most Samsung devices come with the Google Play Store, compelling Google to share WebAPK minting will also fix this issue on Samsung devices.

8. Why Google’s Minting Server?

Why should DMA enforcement obligate Google to open its WebAPK Minting APIs, rather than another remedy which does not rely on Google services?

In order to avoid invoking side-loading warnings, WebAPK Apps are signed with keys pre-trusted by the device. The only set of widely trusted keys in the Android ecosystem are Google’s, and Google will (quite rightly) object to any solution that mandates access to those keys by competitors. To provide them would undermine user security, and OWA is interested in improving security, not harming it. Relying on Google’s existing service side-steps this security dilemma while allowing access to an important feature in a timely way.

Alternative solutions that create a new “local minting” process for trusted browsers have the downside of long time-scales. New OS-level APIs will not become credibly available for multiple years due to the fragmented nature of the Android ecosystem.

Lastly, the code for WebAPK shell applications is already Open Source, enabling competitors to quickly build and submit their own “template” WebAPK shells. Much time has already been lost to stalling by Google, and OWA feels strongly that opening Google’s WebAPK Minting API at a minimum to competing trusted browsers is the fastest, most effective solution given today’s fragmented Android ecosystem.

9. Remedy

The remedy in this case is simple. Google at a minimum needs to identify trusted browsers and provide them the ability to properly install Web Apps using the same WebAPK Minting APIs that Chrome has enjoyed access to since 2017.

Given all the technology is already in place, **this solution could be implemented within weeks.**

This API should be restricted to dedicated browsers, as it could be possible for non-browser apps to abuse this API. Clear policy will enable browsers to benefit from competition with Chrome, and will enable Google to police access in conformance with publicly available guidelines.

The DMA may need to adjudicate in cases where Google has refused to grant access or revoked access to make sure commonsense, fair and evidence based processes are being followed or in the event where there is an industry standard for trusted browsers, delegate to that procedure.

10. WebAPK Install Data

Google should not be allowed to use the data collected via WebAPK minting in any way that contravenes Article 6(2) including aggregated or non-aggregated data.

One reasonable purpose of collecting limited aggregated data is to compare the urls of the Web Apps installed by a particular browser to a list of malicious URLs, this could be used to spot abuse of this API by a malicious or incompetent browser.

Our preference would be that Google publicly publishes the aggregated data so we can measure the success of Web Apps and implements proper information firewalls internally so that this data is not used in competition with its business users.

12. Toward A Brighter Future

Restriction on WebAPK API access and backup/restore for Web Apps are clear instances of behavior prohibited under the DMA. Google can and should provide WebAPK functionality to other trusted browser vendors, and should bring parity to Web Apps and Native Apps in the context of system backup and restoration.

This will allow browser vendors on Android to compete fairly in the provision of Web App features. This is important as it is competition that drives the Web forward and it will prevent Web Apps from being broken on rival browsers.

This is good for consumers as Web Apps are one of the only interoperable and open distribution systems for app development. This combined with the other proposed remedies will lead to cheaper, higher quality, more open and interoperable software for EU’s consumers.

Competition, not walled gardens, leads to the brightest future for EU’s consumers

13. Open Web Advocacy

Open Web Advocacy is a not-for-profit organization made up of a loose group of software engineers from all over the world, who work for many different companies and have come together to fight for the future of the open web by providing regulators, legislators and policy makers the intricate technical details that they need to understand the major anti-competitive issues in our industry and potential ways to solve them.

It should be noted that all the authors and reviewers of this document are software engineers and not economists, lawyers or regulatory experts. The aim is to explain the current situation, outline the specific problems, how this affects consumers and suggest potential regulatory remedies.

This is a grassroots effort by software engineers as individuals and not on behalf of their employers or any of the browser vendors.

We are available to regulators, legislators and policy makers for presentations/Q&A and we can provide expert technical analysis on topics in this area.

For those who would like to help or join us in fighting for a free and open future for the web, please contact us at:

Email	contactus@open-web-advocacy.org
Web / Web	https://open-web-advocacy.org
Mastodon	@owa@mastodon.social
Twitter / X	@OpenWebAdvocacy
LinkedIn	https://www.linkedin.com/company/open-web-advocacy

14. References

Web APKs
https://web.dev/webapks/

Google introduces WebAPK minting
https://9to5google.com/2017/02/02/chrome-57-progressive-web-apps-features/

Google stating they will share WebAPK minting with third party browsers
https://web.dev/articles/webapks#:~:text=We%20are%20working%20on%20it.%20We%20are%20committed%20to%20making%20this%20available%20to%20all%20browsers%20on%20Android%20and%20we%20will%20have%20more%20details%20soon.

What is a WebAPK
https://web.dev/articles/webapks

Digital Signature definition
https://en.wikipedia.org/wiki/Digital_signature

Browser Vendors asking for access to WebAPK minting
https://bugs.chromium.org/p/chromium/issues/detail?id=1243583

Mozilla's study on choice screens
https://research.mozilla.org/browser-competition/choicescreen/

Google is a supporter of the Open Web Foundation
https://en.wikipedia.org/wiki/Open_Web_Foundation#:~:text=According%20to%20its%20web%20site,Google

Samsung announcing WebAPK for Samsung Browser
https://medium.com/samsung-internet-dev/new-year-new-samsung-internet-b74f282e4429