In-App Browsers

Subverting Competition, User Privacy & Choice

VERSION 1.1

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

1. Table of Contents

2. Introduction

Companies such as **Facebook** have been undermining **consumer privacy**, **browser competition** and the open and free web ecosystem by exploiting lax policies by mobile OS vendors. These policies allow companies to deprive users of browser choice at an industrial scale. Facebook hardly needs to track users through cooperation with websites when it can simply replace the user's choice of browser and monitor every tap, scroll and link from it's array of ubiquitous apps.

Google similarly subverts user choice of browser through its home-screen search-box on Android, using its control of the OS to capture users. Apple, although it claims to care about privacy, is complicit in allowing Facebook's abuse of users by not ensuring apps respect a user's browser choice.

Browser choice is the bedrock of competition on the Web and is what pushes companies to compete on privacy, performance and functionality. While apps are allowed to silently undermine this choice by shipping and forcing their own internal built in browsers on the user without asking, that competition along with the users privacy preferences and extensions is discarded harming not only the user but the entire open, free and competitive ecosystem that pushes the Web forward.

To understand these issues is technically complex even for web-developers which is part of the reason it hasn't received attention. Essentially Facebook has worked around the "problem" of users choosing browsers with better anti-tracking controls by removing that choice in a large fraction of website visits. It should be the job of the OS vendors to prevent this from happening.

To protect users and competition, regulators need to step in and require OS vendors to enforce and respect a user's choice of browser across all applications and to ensure a level playing field for any app that wants to take on the role of a web-browser. If a user has chosen a browser, all apps must use that browser for visiting websites.

In-App browsers present a risk to privacy, security, choice and competition

A user's choice of browser should always be respected by the operating system and all apps on that operating system.

3. What is an In-App Browser?

Most people are familiar with the concept of a browser. It's an application that opens and displays web content when users click links, whether that be in an email, sms or a document. An example of what we'd consider a browser is something like Safari, Firefox, Chrome, Edge, Brave or Vivaldi.

Applications that are not browsers often include what's called an In-App Browser ("IAB"). These systems give app developers an easy way to display web content from third parties within their applications, without "losing" users that would otherwise be taken out of their apps and to their browsers. They can do this by utilising the system-provided "webview" rendering engine to load content. Other uses of webviews include displaying first-party web content (e.g., the contents of an email in a mail application) or cooperating third-party content (ads). In-app browsers are different from these uses of webviews in that they are used to display content from other developers who have not consented to having their web pages loaded in a non-browser environment.

In-App Browsers become a problem when apps use them to display **third-party content**, i.e. content they have no involvement with, as this undermines a user's choice of browser and hobbles the functionality of websites rendered in these systems.

This also removes choices the user has made about privacy, security and functionality within their default browser. If a user's browser choice is not respected then this harms competition and often can harm functionality and the web ecosystem in general.

As a general matter, user's choices in their default browser should impact how links they visit from within apps are handled, and developers building third-party content should not be forced to consider the ways in which IABs remove functionality in largely untestable ways.

There are many other pathways to accessing web content. For example:

- Receiving a link in a chat or social media program
- Tapping a link from a story in social media feeds
- Receiving an email within an app that contains a link to a website
- Users posting links in moments / feeds or on forums

Once the user clicks or taps on a link, there are a number of different actions that can happen depending on the app the user is using and how the app and operating system handle links. This document will outline what happens currently on operating systems, the issues and then proposed fixes.

4. Definitions

For the purposes of explanation, we're going to split browsers into several definitions.

- Default Browser (Default Browser)
- In-App Remote Tab Browser (Remote Tab IAB)
- In-App WebView (WebView IAB)

Although WebViews and In-App Browsers (IAB) are terms that are used, the nomenclature, Remote Tab IAB and WebView IAB is specific to this document and is not yet used across the industry.

4.1. Default Browser

Default Browser - A user's default browser.

When a user taps a link, the link's target website will open in the user's current default browser separately from any app.

For example, when a user receives a message containing a link and they tap on it, the focused application will change from their SMS application to the Default Browser (e.g. Firefox) and the website will load in that browser outside of the app.

This is the default behaviour of operating systems since the 90s and preserves user choice in browsers. It also ensures the privacy and security settings of their browser (including extensions) are applied when browsing sites from these links. This default behaviour enables browser competition and respects user choice.

Mobile operating systems do not always respect this choice.

4.2. In-App Remote Tab Browser

Remote Tab IAB - A tab from a real browser, but shown inside an app.

This can occur when a user taps a link and the website loads within the current application (i.e. it does not switch to a browser application) but the site is still rendered using the user's chosen (default) browser. Twitter for Android implements this pattern using Chrome Custom Tabs ("CCT") in the default configuration. It is important to note that while CCT contains the word Chrome, under default settings it will load the users' default browser even if it is not-chromium, for example Firefox (Gecko).

User's default browser choices -- whether Firefox, Edge, Chrome, or Opera -- are respected when the user's default browser is "projected" through a Remote Tab IAB. This behaviour strikes a balance between developer interests in not "losing" users to external applications (browsers) and the user's agency in choosing and configuring their own browser.

By default this does not undermine user choice, however on Android there is an option in CCT, which is the component that is used to enable Remote Tab IAB to specify a single engine. This can be used to undermine user choice like in the Android Google Search App. On iOS this always uses Safari whether or not it is the user's default browser, disrespecting the choices of users that select different default browsers.

4.3. In-App WebView Browser

WebView IAB - A OS-Provided Component for Rendering Web Content

Many apps provide an IAB based on the OS-provided webview component. Historically, this was grounded in reasonable motivations by app authors, but as Remote Tab Browsers have become available, the problems with WebVIew IABs have become evident.

Apps that implement WebView IABs:

1. Do not respect the user's choice of browser, undermining competition

2. Reduce functionality relative to "real" browsers. WebViews are not designed to be drop-in replacements for browsers, and significant work is needed to fill in the gaps; work that many apps do not put in.

3. Rely on the operating system to update the WebView component, potentially exacerbating security risks that real browser vendors do work to mitigate.

4. Can MITM (Man in the middle) connections to third parties, which means they can intercept all data sent and received to the website without the User's knowledge or consent. This creates silent tracking/privacy & security risks.

5. Can monitor every click, tap, input and interaction with a WebView

6. May not mitigate known security risks in outdated system WebViews (e.g. on older devices), putting users at high risk compared to loading websites in real browser

7. May fail to implement key features (e.g., Push Notifications), despite the underlying WebView providing APIs that would allow it.

   WebViews are not fully functioning browsers and applications that use them when Remote Tab IAB systems are available are undermining user' choice in browsers. They also may damage the wider Web ecosystem by causing bugs and removing functionality critical to the content they load.

   A variant of System WebView IABs are IABs built on application-provided browser engines (e.g., Mozilla's GeckoView, which can be embedded in many native apps).

5. Android and iOS Current Implementations

5.1. ANDROID

Apps can either:

1. Open links in the user's Default Browser
2. Invoke a Remote View IAB that uses either a User's Default Browser (the default for CCT), or a hard-coded browser as the Google home screen search box does, always calling into Chrome, undermining user choice in browser
3. Implement a WebView IAB

At the moment, Android apps have infinite flexibility. We believe the choices that undermine user choice as well as privacy and security need to be restricted or removed on Android to encourage healthy competition, both between browsers as well as between web apps and native apps.

The In-App Remote Tab Browser on Android uses "Chrome Custom Tabs" ("CCT")[1].

5.2. iOS

Apps can either:

1. Open links in the user's Default Browser, all of which must use Apple's WebKit engine thanks to section 2.5.6 of Apple's App Store Review guidelines
2. Open in the SFSafariViewController Remote Tab IAB. This Remote Tab IAB *always* invokes Safari, undermining browser choice by users
3. Open using a Webkit WebView (WebView IAB) using the specific locked WebView and version of Webkit provided by iOS

There are no other options.

[1] Although the name "Chrome Custom Tabs" confusingly contains the word "Chrome", by default it opens a Remote Tab to the user's default browser engine. CCT, minus the ability to hard-code a specific browser, is a model for app developer and user choice being respected.

6. Recommendations

Default Browsers and Remote Tab IAB should be the only options for third party content. It should not be technically possible for an app to access third-party content outside of either being a Default Browser or a Remote Tab IAB.

In-App WebView's should be limited to work only with:

1. First Party Content

- Content for a developers app that the developer made

2. Second Party Content

- Content from a second party that the first party has a strong business relationship with (i.e. adverts)

3. Building Browsers

- As a building block for building browsers

These restrictions should be enforced by the OS. Regulatory oversight and policy should require that:

1. **Third Party Content should open only via the User's Default Browser**
   In-App WebViews should be strictly limited to content provided by that app's publisher and direct collaborators (including adverts). OS and app store review policy should prohibit apps from loading third-party content in WebView IABs. All third party content should be loaded by the users' Default Browser or through a Remote Tab IAB.

2. **Applications should only be able to make themselves available to be set as the users Default Browser if they are a Browser**. More specifically this should only be available to Applications whose **primary purpose** is to browse third party content on the Web, complete with all the typical trappings of a browser such as url address bar, back/forward buttons, refresh button, search bar, multiple tabs etc. This provision should be enforced by the OS and app store review policy. While this sounds obvious, this is important to stop gaming by unscrupulous companies. For example popular Social Media Apps and messaging services should **not** be able to declare themselves a browser.

3. **User's can choose Remote Tab IAB or their Default Browser per App**
   All Apps should be enforced by the OS to prompt users whether they would like to open links in an **Remote Tab IAB** or in their **Default Browser** (i.e. switch app to their default browser). Apps can not override this preference.

4. **Websites can opt for Default Browser only**
   Websites should be allowed to indicate to apps that they only wish to be opened in the Default Browser (for instance, through an HTTP header or HTML markup) and this must be respected / enforced by the operating system.

5. **OS Search Widgets must respect Browser Choice**
   The "Android Google Search App" and similar operating system level widgets must respect the users choice of browser.

6. **Remote TAB IAB should not have an option of specifying individual engines**
   When loading third-party content, application developers should not be given the option to override the user's selection of browser.

7. **OS Vendors Should Allow Centralised Opt-Out of IABs Entirely**
   Operating systems should provide a central toggle that disables the use of IABs for third-party content, and instead always takes users to their default browser when they tap on links within apps.

8. **No Direction to Specific Browsers**
   Apps (such as gmail) should not offer specific functionality to open in a particular browser. They should only offer to open content either in a Remote Tab IAB or the Default Browser. This should be enforced by the OS.

9. **First Party Content** and **Second Party Content** would need a technical mechanism by which to opt-in to being rendered within an app. Careful consideration would need to be given to this mechanism to ensure it's strictly for first and second party content.

A user's choice of browser **should always be respected**
by the **operating system & all apps**

7. Further Reading

The authors would like to note that this submission is based on the following articles which provided most of the technical understanding and ideas to produce this submission. These add additional technical and background detail that is worth reading for a deeper understanding.

Hobson's Browser
How Apple, Facebook and Google are undermining user choice in browsers
https://infrequently.org/2021/07/hobsons-browser/

Alex Russell: Hobson's Browser: How Browser Choice Died In Our Hands
https://www.youtube.com/watch?v=6FaI9zQtgvM

8. Open Web Advocacy

Open Web Advocacy is a loose group of software engineers from all over the world, who work for many different companies who 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 came together with the hope of creating a better world since no major company has been pushing for this change in recent years.

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
Twitter	@OpenWebAdvocacy
Web	https://open-web-advocacy.org