0.0.1 Abstract

I made my own research blog about Firefox and Chromium, pulling from low-level source repositories, security documentation, and the actual online available information on the Web. research detailed information that up to date and accurate improvements, post accurately detailed how dual-engine attack surface argument (Firefox requires Chromium WebView on Android) is a platform architectural constraint, not a Firefox deficiency, as well research also detailed how individual threat model is more integral to overall security than default sandboxing as well as exploit mitigation differences that end up being arbitrary if indeed exploits tend to use social engineering Firefox completely outclasses Chromium at network-layer privacy isolation, but it takes a second place when it comes to trapping a weaponized mobile exploit, it is improving, In my blog, i was accused of "LLM-generated biased text", i don't use LLMs for writing my Text(because i did not write in personal style but rather in research standard), I however do use multiple different styles of writing(because of sheer amount of accusations i did rewrite blog in "human language" and outlined it in updates, which was further used to jusitify AI accusations), it being Formal Academic Style(based on APA Standard), as well as my own natural style, which i am forced to write it because of sheer amount of toxic backlash, but i will indeed keep writing a Blog, given a research of how different browsers have different strategies in Sandboxing ### The "Dual-Engine" Attack Surface is a Platform Law, Not a Firefox Flaw

This is an argument i haven't seen addressed, at all(just disregarded as inaccurate with no evidence or reasoning) One of the most common arguments weaponized against Firefox on Android is that it introduces a dangerous "dual-engine" attack surface. The claim is that by installing Firefox, you are actively exposing your device to two major codebases at the same time: `GeckoView` and the native Chromium-based `Android System WebView`. [1]

My research accurately detailed how this dual-engine argument is a platform architectural constraint indeed, not a Firefox deficiency in true and valid sense.

System Developers giving preferential treatment to Chromium and it's Webview reinforces monopoly that worsens security. [2] Firefox did not introduce that attack surface, the Android platform however did. Blaming Firefox for the passive existence of a mandatory system component is a fundamental logical error. ### Threat Modeling Matters More Than Arbitrary Mitigations

My research also detailed how an individual's specific threat model is far more integral to overall operational security than default sandboxing configurations or exploit mitigation differences. In the real world, these highly technical defensive differences end up being completely arbitrary if indeed exploits tend to use social engineering.

If an adversary compromises a user via phishing, malicious APK sideloading, or basic credential theft, the depth of a browser's JIT compiler sandbox means absolutely nothing. The attack bypassed the browser engine entirely.

When evaluating the two browsers where the technical boundaries *do* matter, it comes down to a deliberate architectural trade-off:

- **Firefox completely outclasses Chromium at network-layer privacy isolation.** [3] Features like Total Cookie Protection [3] automatically wall off storage and tracking graphs into independent "jars" for every single first-party domain, natively preventing corporate surveillance and cross-site fingerprinting without breaking client-side JavaScript APIs.
- **However, it takes a second place when it comes to trapping a weaponized mobile exploit.** On Android, Firefox lacks kernel-level process containment (`android:isolatedProcess`) [4] and does not possess an intra-process sandbox for its SpiderMonkey JIT engine equivalent to Chromium's V8 Sandbox. [5]

Crucially, **it is improving.** Mozilla is actively working through these historical mobile architectural deficits, recently pushing Project Fission (site isolation). [6] It is steadily playing catch-up to harden its post-compromise net. ### The Conclusion

Security is not a single, linear scoreboard where one browser wins a universal trophy. It is a choice between two entirely different engineering paradigms. Chromium builds an incredibly complex web API surface (exposing high-risk hooks like WebUSB and Direct Sockets) [7] but wraps it in state-of-the-art post-compromise containment. Firefox radically reduces the pre-compromise attack surface by omitting those complex APIs entirely and writing outer parsing layers in compile-time memory-safe Rust (research groups pointed out that Chromium also invests in memory-safety as well as Rust components, Chromium is indeed memory-safe), [8] while explicitly optimizing for user privacy.

An objective assessment looks at what you are actually trying to defend against, rather than blindly bowing to the authority of a specific dev team's preferred threat model.

0.1 Intention

I, the Author, was accused of being biased towards a browser, which isn't worth dignifying as a response, I discovered new information regarding browser security, as Data Analyst, I incorporated it into my research blog, However, something that was distasteful(I am writing this in "human" language), almost purposeful and targeted misunderstanding which I feel like I have to demostrate, how some research groups view skepticism, frame it as a response to the bias towards a browser, and actively encourage inline groups to "counter" it

0.1.1 References

[1] GrapheneOS, "Usage: Web Browsing." [Online]. Available: https://grapheneos.org/usage#web-browsing.

[2] D. Geer et al., "CyberInsecurity: The Cost of Monopoly," CCIA,
2003.

[3] Mozilla, "Total Cookie Protection," Mozilla Security Blog. [Online]. Available: https://blog.mozilla.org/security/2021/02/23/total-cookie-protection/.

[4] Android Open Source Project, "Isolated Process," Android Developers. [Online]. Available: https://developer.android.com/guide/topics/manifest/service-element#isolatedProcess.

[5] V8 Sandbox Design Document. [Online]. Available: https://docs.google.com/document/d/1FM4fQmIhEqPG8uGp5o9A-mnPB5BOeScZYpkHjo0KKA8.

[6] Mozilla Wiki, "Project Fission." [Online]. Available: https://wiki.mozilla.org/Project_Fission.

[7] Chromium Project, "Direct Sockets API." [Online]. Available: https://wicg.github.io/direct-sockets/.

[8] Chromium Project, "Rust in Chromium." [Online]. Available: https://security.googleblog.com/2023/01/supporting-use-of-rust-in-chromium.html.