Investigating Mobile Privacy: Browser Fingerprinting on Privacy-focused Mobile Operating Systems

Noah Zoarski

Purdue University CNIT

CNIT32200: Research Methodology and Design

Dr. William Ledbetter

November 27, 2023

Table of Contents

Introduction

As the implications of poor security and data privacy are becoming increasingly apparent in users' everyday lives, the value gained by researching the mechanisms that maintain regular user privacy cannot be overstated. Browser fingerprinting is a sophisticated online tracking technique that leverages a tremendous amount of unique data points captured from a user's browsing session in order to identify them online, typically for advertising and content delivery purposes. Characteristics such as, but certainly not limited to, operating systems, windowing systems and window resolution, browser types and engines, installed fonts, timezone, and even battery percentage are combined to create a unique 'fingerprint' of a user. Unlike cookies, the covert and insidious nature of these methods allows fingerprinting to exist in a regulatory gray area and operate without the user's knowledge or consent, eluding even the most stringent legal definitions of privacy like the CPA and GDPR.

This research paper aims to recognize an underrepresented and highly relevant user base not previously considered in publications on browser or device fingerprinting: GrapheneOS and LineageOS users. Graphene and LineageOS are privacy-forward forks of the Android codebase that come bundled with their own custom browsers preinstalled. Laperdrix et al. have already demonstrated how browsers on mobile appliances like the Apple iPhone and Android devices prove more resilient to modern fingerprinting techniques than their desktop counterparts due to the homogeneous environment and minimal configurations allowed by Apple or outright reliance on Google's Chromium–an established privacy browser when configured appropriately. While most mobile users will continue to use browsers such as Apple Safari and Google Chromium, it is crucial include privacy-forward distributions like GrapheneOS and LineageOS in our research. To not only underscore the privacy-committed by ensuring their safety and security, but also to dignify their contribution in driving forward privacy enhancements that have the potential to benefit broader digital populations.

GrapheneOS is of particular focus in this study for its ubiquity and reputation among fringe privacy-conscious users. LineageOS, under less of a limelight, will also be examined for similar reasons. While both projects promise privacy enhancements to the overall Android codebase, the extent to which these mechanisms are implemented has not yet been well defined.

While existing research suggests improved fingerprinting resilience among mobile browsers, the privacy-focused operating systems we seek to investigate might not philosophically align with proven anti-fingerprinting models. Unlike the mobile browsers that proved successful in Laperdrix et al. and Hraška's 2016 studies, Graphene and LineageOS offer greater user autonomy in system configuration, which could imply a more unique fingerprint, making the operating system's default browser a prime target for research.

With respect to the upstanding publications that came before this, the proposed study is largely replicative. Looking to implement novel technologies to measure more modern characteristics for fingerprinting, technologies such as CreepJS and FingerprintJS will be connected to, instead of self-hosting a service or using a script to fingerprint browsers to take measurements, eliminating a potential need for software development. Furthermore, Unblocked Web will be deployed to aid in the controlled deployment of the browsing environments, thus eliminating the need for a Google Pixel cell phone. Furthermore, relevant formulae, such as Shannon's Formula for entropy, will be applied similarly to Hraška's 2016 study.

Flawedworld, an Android developer and significant contributor to the GrapheneOS codebase was contacted by the author regarding an outstanding issue on GitHub, and was incredibly helpful in sharing resources on fingerprinting and browser technologies. Certain suggestions from Flawedworld will be implemented in the study, such as using CreepJS, FingerprintJS, and Unblocked Web, which will undoubtedly contribute to the study's validity and reproducibility.

By narrowing the focus of the study to privacy-focused mobile distributions, this research seeks to thoroughly examine the extent of these operating systems' privacy capabilities. While these operating systems are lauded for their privacy and popularity among enthusiasts, little evidence has shown that the Vanadium or Jelly browsing experience is actually any more private than Android's out-of-the-box. In fact, as far as we know, it could be worse. The findings of this study are intended to inform the privacy-concerned decision when settling on a mobile operating system by continuing to explore the extent of modern fingerprinting capabilities.

Literature Review

A breadth of publications since 2010 have contributed substantially to the space of web browser security and fingerprinting. A 2010 study from an EFF article published by Peter Eckersley analyzed 470,161 browser fingerprints collected using the Panopticlick website hosted by the EFF themselves. This investigation explored the evolving impact of modern web technologies on browser fingerprinting. The study found that while advancements in accessibility have certainly improved user experience as a whole, accessibility often comes at the cost of user privacy. Of particular note, the study provided the first extensive analysis of fingerprints from mobile devices, revealing that an astonishing 81% of mobile fingerprints were entirely unique. The study emphasized the significance of HTTP headers and notably, the role of the HTML5 canvas element in device identification, as well as the role of cookies, IP addresses and supercookies when trying to correlate user data. Utilizing the HTML5 canvas to create unique fingerprints is a theme that Hraška further elaborated upon in his Master's Thesis.

Laperdrix et al. (2016) also expanded upon the IEEE article, providing a comprehensive analysis of fingerprints collected from mobile devices, revealing the unique challenges when fingerprinting mobile devices.

Hraška's extensive research from about the same time unveiled the granular methods and features used to uniquely identify users. By testing various combinations of browser features, Hraška identified the smallest subset of browser features that produce substantial entropy. Entropy, in this context, refers to the amount of uncertainty or randomness associated with the browser's features, making it a measure of a user's uniqueness. To elaborate, every bit of entropy doubles the amount of distinguishable user states. Thus, an entropy of 14.5 bits means there are approximately 27,000 unique browser states. So, the higher the entropy, the greater the potential for unique identification. The study found that by employing just nine specific browser features, a stand-out entropy measure of 16.5 bits can be reached, suggesting that even minimal data exposure can lead to a high level of user identifiability. And by employing just three features–date and time, user-agent, and window resolution–an entropy of 14.2 bits can be measured. Better yet, Hraška outlined all the necessary mathematics used for normalizing datasets and calculating entropy that will be employed in this replicative study.

It is important to note Laperdrix' and Hraška's observation of mobile devices being considerably more resistant to fingerprinting than desktops. Hraška concluded that mobile devices exhibited less entropy, and thus fewer unique fingerprints. This could be attributed to the homogeneous nature of mobile device ecosystems, especially within certain jurisdictions like Apple's "Walled-Garden". Apple's ecosystem is highly uniform, with many users employing similar configurations and software, thus reducing variations and consequently, entropy. This uniformity inadvertently protects users from fingerprinting since the fewer discrepancies there are between devices, the harder it becomes to single out any one in particular. This particular mechanism of privacy is incidentally enhanced by the fact that Apple users have greater power in numbers.

Users have recently flocked to platforms like Graphene and LineageOS with the anticipation of escaping services that are suspected to harvest data and invade privacy. Their trust hinges on the expectation that the OS, even if not primarily designed for web browsing, should provide a holistic private experience in its operation, which includes browsing. After all, in an age defined by the internet, it is reasonable for users

to expect that a smartphone—regardless of its primary use-case—would facilitate a private web browsing experience.

The Lineage and GrapheneOS projects, like many other open-source projects, face unique challenges in development. Besides just a lack of funding, one inherent challenge arises from the projects' niche user bases, which are primarily composed of technical hobbyists and privacy enthusiasts, thus lacking the power in numbers seen with platforms like the iPhone. Furthermore, technical users and those who prefer open-source software tend to exercise autonomy in their decision to deploy their own hardware and software, which could contribute to even more entropy.

Looming over the prospect of Vanadium's claim to privacy, GitHub Issue #181, raises significant concern regarding the exposure of unique hardware IDs. The issue describes a vulnerability where devices might leak a distinct fingerprint due to the exposure of hardware-related information. While a solution to hardcode WebRTC `device_id_salt` has been proposed to mitigate this issue (Flawedworld, 2023), it has not yet been implemented. This unresolved challenge is of particular relevance to our study, as it suggests that the entropy of browser fingerprints for GrapheneOS users could be considerably higher than its mainstream counterparts, and, in the worst case, render Vanadium ineffective for private browsing.

When a software exposes hardware IDs, it reveals far more unique user data than canvas fingerprinting. Like a MAC address, such hardware identifiers are low-level and persistent unless changed, which should triage the hardware ID issue with priority in the Vanadium backlog. The methods outlined in publications prior are certainly capable of measuring hardware IDs, and should be reapplied to audit the Vanadium and Jelly browsers. With mainstream browsers already implementing countermeasures like fingerprinting alerts, it seems like GrapheneOS has sizable ground to cover.

As the digital ecosystem evolves, the methods of fingerprinting and fingerprinting prevention evolve concurrently. Platforms like GrapheneOS stand at the vanguard of the privacy movement, and are certainly expected to maintain the safeguards vital to user privacy. Considering the scope of fingerprinting

capabilities, it becomes clear that GrapheneOS is due for a security audit. Future research, especially focusing on platforms like GrapheneOS's Vanadium, is not merely beneficial—but imperative; not only to the technical user, but to those who actually have something worth hiding.

Methodology

The proposed research methodology is designed to be inexpensive and easily replicated. By using an array of advanced web technologies such as Unblocked Web, CreepJS, and FingerprintJS, measurements and emulation will be capable of running almost headless, making the hardware requirements little to none. The research methods will closely mimic Hraška's for a comprehensive and familiar evaluation of Vanadium's capabilities against browser fingerprinting, only replacing Hraška's script with CreepJS and FingerprintJS to minimize project scope.

Utilization of Unblocked Web: The primary function of Unblocked Web in this study is to emulate a diverse range of browser environments. By simulating various operating systems and browsers, we aim to create a comparative framework within which Vanadium's fingerprinting resistance can be assessed against other popular browsers. This emulation can run headless and helps isolate the test environment. As a result, data processing and normalization will be very straightforward.

Data Collection with CreepJS and FingerprintJS: CreepJS and FingerprintJS are employed to gather detailed fingerprints from each emulated browser environment. These tools are still supported, and are capable of extracting a wide array of data points, including, but not limited to, hardware IDs, canvas prints, and user-agent strings. The use of websites as opposed to a local script will allow for a simple controlled and systematic collection of fingerprint data as the emulated browsers from Unblocked Web access these sites.

Analysis and Interpretation

The fingerprint data collected from Unblocked Web, CreepJS, and FingerprintJS will be of numerous forms, each capable of attributing a unique amount of entropy to a browser. Initially, the fingerprints will be cataloged into an organized structure where each entry corresponds to an observed set of browser characteristics. These include, but are not limited to, items like hardware IDs, canvas prints, and user-agent strings. This database will then become the foundation for a comparative analysis, wherein Vanadium's fingerprinting data can be compared against the fingerprints from other browsers deployed using Unblocked Web. Statistical methods will be employed to identify patterns and anomalies. Key metrics such as the frequency of unique fingerprints and the prevalence of shared attributes across different browsers will be meticulously computed. This analytical approach is integral to discerning the relative strength of Vanadium's fingerprinting resistance. It enables a nuanced understanding of how the browser's privacy features perform when exposed to the same tracking vectors as other browsers.

To quantify the distinctiveness of each browser's fingerprint gathered from our experiment using Unblocked Web, CreepJS, and FingerprintJS, we apply Shannon's entropy formula. The process begins by cataloging the unique fingerprints identified across the dataset. Each unique fingerprint, denoted as x_i, represents a specific combination of browser features, such as hardware IDs, canvas prints, and user-agent strings.

The probability mass function $P(x_i)$ is then derived from the frequency of each unique fingerprint in the dataset. If we have observed a particular fingerprint x_i in f_i out of N total fingerprints collected, the probability $P(x_i)$ is calculated as f_i/N. This frequency-based probability reflects the likelihood of encountering that specific browser fingerprint in the population of collected data.

Subsequently, the entropy $H(x)$ of the entire system of fingerprints is calculated using the formula:

$$H(X) = -\sum_{i=1}^{n} P(x_i) log_b P(x_i)$$

In practice, we execute this calculation programmatically, where each $P(x_i)$ is determined by the experimental data, and the sum is computed over all unique fingerprints. This calculation yields the entropy $H(x)$ in bits, which provides us with a metric for the average amount of information—or unpredictability—present in the distribution of browser fingerprints.

By applying this mathematical approach to our experimental data, we are able to assess the fingerprinting resistance of Vanadium and Jelly browsers quantitatively. The entropy measures serve as a critical indicator in our analysis, revealing the effectiveness of browsers' privacy-preserving mechanisms.

Findings and Implications

This research anticipates uncovering distinguishable patterns of fingerprinting resistance across various browser environments and configurations, with a particular focus on GrapheneOS's Vanadium. The expected findings should reveal how the entropy of Vanadium's and Jelly's fingerprints compare to each other and to those of mainstream browsers. Given the unresolved hardware ID leak on GrapheneOS, we hypothesize that Vanadium may exhibit a higher entropy level than browsers on more conventional mobile operating systems. This potential outcome would suggest a need for GrapheneOS developers to consider additional privacy-preserving mechanisms, such as the proposed but not yet implemented hardcoding of WebRTC `device_id_salt`.

The implications of these expected findings are several fold. First, should the hypothesis of higher entropy in Vanadium's fingerprints hold true, it would underscore a critical vulnerability in the browser's

capacity to protect against user tracking and identification. This would contradict the prevailing assumption that privacy-focused mobile operating systems naturally offer superior anonymity. Still, Vanadium's entropy profile should be considered with and without the hardware ID measure, so that measurements will reflect the efficacy of Vanadium with and without a WebRTC `device_id_salt`.

Furthermore, the analysis may inspire a broader discussion on the necessity and feasibility of salting hardware IDs as a standard privacy practice. Such discourse could pave the way for more robust privacy protections, not just for fringe users but for the digital ecosystem at large. It also serve as an empirical foundation for privacy advocacy, emphasizing the importance of developing and implementing advanced privacy features.

Bibliography

Browser fingerprinting - Masters Thesis. (2018). https://www.virpo.sk/browser-fingerprinting-hraska-diploma-thesis.pdf

Eckersley, P. (2010). How unique is your web browser? *Privacy Enhancing Technologies*, 1–18. https://doi.org/10.1007/978-3-642-14527-8_1

GrapheneOS. (2022, May 11). *Hardcode WebRTC device_id_salt for built in hardware on grapheneos devices · issue #181 · grapheneos/vanadium*. GitHub. https://github.com/GrapheneOS/Vanadium/issues/181

Laperdrix, P., Rudametkin, W., & Baudry, B. (2016). Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. *2016 IEEE Symposium on Security and Privacy (SP)*. https://doi.org/10.1109/sp.2016.57

Xu, Q., Zheng, R., Saad, W., & Han, Z. (2016). Device fingerprinting in Wireless Networks: Challenges and Opportunities. *IEEE Communications Surveys & Tutorials*, *18*(1), 94–104. https://doi.org/10.1109/comst.2015.2476338

Zoarski, N., & flawedworld. (2023, October 27). Personal communication. personal.