Technical analysis of browser fingerprinting techniques based on FingerprintJS

James Bergfeld
Technical University Munich
Munich, Germany
j.bergfeld@tum.de

Samuel Scheit
Technical University Munich
Munich, Germany
tum@samuelscheit.com

I. Browser fingerprinting

A. General

Websites use browser fingerprinting to create a unique identifier of each website visitor by collecting data about the visitor's device and browser settings and combining them into a unique "fingerprint."

The aim is for website operators to identify users across multiple website visits without them having to actively accept cookies or log in with their user accounts.

B. Advantages

The purpose is to create a detailed profile of each user to display personalized content, serve advertising or analyze user behavior. This can be used both to improve the user experience and to detect fraudulent activity.

C. Disadvantages

In order to create a unique browser fingerprint, extensive information about a user's devices and browser settings must be collected. However, this violates the user's privacy unless they have explicitly agreed. Especially since there is no way to opt out of fingerprinting and the data can be used to track users across multiple websites. This allows comprehensive profile of a person's online activities to be created and conclusions to be drawn about a person's identity and behavior.

D. Relevance

Because website operators require unique user profiles, even without users' consent, to provide personalized content and to analyze user behavior, browser fingerprinting has become an important tool. This is evidenced by the fact that 30.6% of the top 1k websites in the Alexa ranking use fingerprinting techniques. [1]

Since the majority of all browsers deactivate third-party cookies by default in the future[1], or need explicit consent to use third-party cookies, browser fingerprinting is a significant alternative to identify users across different websites.

E. Application

In order to assign a unique identity or "fingerprint" to each user, various details are collected via the browser. For example, a combination of rare fonts, a specific screen resolution, or a specific browser plugin can help generate a unique fingerprint.

JavaScript libraries can be used for this, such as FingerprintJS, which collects a variety of information about a user's browser environment. In the commercial version, FingerprintJS claims to be able to create a 99.5% unique fingerprint. [2]

FingerprintJS is the most popular JavaScript browser fingerprinting library according to npm downloads.[2]

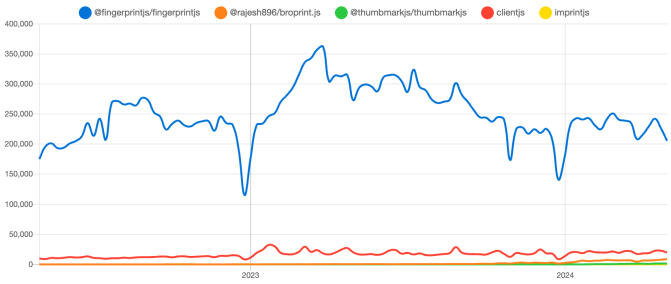

Figure 1: NPM downloads per day, comparison of different JS fingerprinting libraries (as of 2024)

[1] https://developer.mozilla.org/en-US/blog/goodbye-third-party-cookies/

[2] https://npmtrends.com/@fingerprintjs/fingerprintjs-vs-@rajesh896/broprint.js-vs-@thumbmarkjs/thumbmarkjs-vs-clientjs-vs-imprintjs

II. Proposal

A. Project Idea

Technical analysis of browser fingerprinting techniques based on FingerprintJS

In this paper we examine in detail how browser fingerprinting techniques work in practice, from capturing device and browser signatures to generating unique fingerprints. Various aspects are taken into account, such as the use of the browser canvas, the identification of plug-ins and fonts, and the detection of device parameters. [3]

The library referenced for practical implementation is FingerprintJS. We analyze the methods and browser APIs it uses to create fingerprints and compare the accuracy of these fingerprints to the open-source FingerprintJS implementation. [4]

We also examine how robust these techniques are to changes in the browser environment, e.g. software updates and configuration changes.

Additionally, we provide an assessment of the privacy and security risks associated with browser fingerprinting. This includes the ability to identify users across different websites and the effectiveness of privacy protections against fingerprinting techniques.

The results will be documented in the form of a paper that explains how browser fingerprinting techniques work, shows their advantages and disadvantages, and offers recommendations on possible countermeasures.

B. Relevance

The technical implementation of fingerprinting technology based on FingerprintJS is a highly relevant topic because browser fingerprinting is widely used on the web, with an increase in the last few years [1]. FingerprintJS was chosen as the library to be analyzed due to its prevalence as one of the most used fingerprinting libraries. Our analysis helps us to understand how fingerprinting is implemented, outlines possible countermeasures and their respective effectiveness. On the one hand, this allows browser manufacturers and browser extension developers to limit or customize possible interfaces to make browser fingerprinting more difficult. On the other hand, library developers can use the identified techniques to improve and make their own fingerprinting libraries more effective.

C. Research questions

1. How does the FingerprintJS library generate unique browser fingerprints?
2. Which browser properties are examined by the FingerprintJS library?
3. How can parts of the library be replicated to build a browser fingerprinting library?
4. How can these techniques be used or adapted to by other actors such as browser manufacturers, extension developers and library developers?
5. What impact do the identified techniques have on user privacy and security?
6. How reliable are the fingerprinting mechanisms and can FingerprintJS keep up with their promise of 99.5% unique device identification and high temporal fingerprint stability?

D. Methodology

A technical analysis of a [state of the art] fingerprinting suite provides insight into modern browser fingerprinting technology.

FingerprintJS is currently the most widely used browser fingerprinting library [5]. As such it's an [excellent] subject to investigate the implementation of browser fingerprinting in practice.

FPJS offers two fingerprinting solutions: FingerprintJS, an open source library with moderate coverage [of different browser types and configurations] and FPJS Pro, a subscription-based closed source library that uses a greater set of parameters and promises a 99.5% rate of (re-)identification.

Since the source code of the latter isn't accessible, it's necessary to reverse-engineer FPJS Pro in order to draw conclusions about the techniques used to generate fingerprints.

In order to create a stable [testing / reverse-engineering] environment, the JavaScript code is extracted from the demo website and deployed locally.

Using a JavaScript Proxy object[3] a list of API- and function calls can be captured which [grants] and overview of the general behavior of the library. This aids in identifying entry points for later reverse-engineering.

Analysis of the network calls made when running FPJS reveals that fingerprints are generated by collecting data about the browser setup using client-side JavaScript and sending it to a remote server. The server parses the data and computes a fingerprint.

The client payload is serialized before being sent and can't be read in plaintext. Therefore the request needs to be traced back to the caller function that generates the payload to decipher its contents. The browser's internal JavaScript debug-

[3]https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Proxy

ger can be used to inspect the payload in plaintext format before it has been serialized.

The plaintext payload can be used to identify the functions for each parameter. These functions are then analyzed in detail, especially with regard to the browser APIs used and their processing.

The actual fingerprints are generated with API calls to a closed source server. This hinders attempts to reverse-engineer the actual generation process.

Systematically testing different parameters from a collection of various payloads while comparing the uniqueness of the resulting fingerprints. This hints at the Relevance of each parameter to the final fingerprint.

The algorithm used to generate a fingerprint from the parameters sent by the client must be carefully designed to accept small changes that a user agent naturally undergoes during use, all the while clearly differentiating between separate clients.

To gather statistical data about the prevalence of each parameter, a custom fingerprinting library based on findings from the previous steps can be implemented. Conclusions can then be drawn from the data about the unique identifiably of each parameter.

References

[1] Z. S. Umar Iqbal Steven Englehardt, "Fingerprinting the Fingerprinters: Learning to Detect Browser Fingerprinting Behaviors." [Online]. Available: https://arxiv.org/pdf/2008.04480

[2] [Online]. Available: https://fingerprint.com/

[3] N. N. J. P. Konstantinos Solomos Panagiotis Ilia, "Escaping the Confines of Time: Continuous Browser Extension Fingerprinting Through Ephemeral Modifications." [Online]. Available: https://www.cs.uic.edu/~polakis/papers/solomos-ccs22.pdf

[4] [Online]. Available: https://github.com/fingerprintjs/fingerprintjs

[5] A. K. Junhua Su, "Automatic Discovery of Emerging Browser Fingerprinting Techniques." [Online]. Available: https://www.kapravelos.com/publications/fptechniques-www23.pdf