Browser Fingerprinting: Attacks and Applications

Ryan Havens

Tufts University

ryan.d.havens@gmail.com

ABSTRACT

With the rise of big data and artificial intelligence it is critically important, now more than ever, to understand the ways our data can be captured and processed. In this paper I illustrate the dangers of a specific method of data tracking known as “browser fingerprinting” and discuss how browser fingerprinting attacks can lead to a plethora of subsequent issues. I implement and publish a working example of a cross-site browser fingerprinting exploit using nothing more than an open-source JavaScript fingerprinting library, a Python backend, and a MongoDB database.

It is imperative that we understand the risks we face online as a result of poor data security policies in order to preserve digital rights.

INTRODUCTION

Internet tracking has evolved greatly since its humble beginnings. The internet was once a place where users had a sense of anonymity. The notion of digital advertising was restricted to cheap banner ads placed at the top of websites declaring that you, yes YOU, were the 1,000,000th visitor to the site. In those days, websites could hardly even be certain if they had garnered 1,000,000 visitors.

Online advertising grew in prominence as companies realized that their marketing campaigns were more likely to be successful if they focused on a specific targeted audience. This led to the beginning of digital targeted advertising in its most primitive form. The idea was this: given some assumption about the content of the page, (e.g. a fashion blog), ads relevant to that content (such as a sale at Abercrombie & Fitch) were more likely to see higher click-through rate, or CTR. Ads leveraged the expected audience of the page without directly tracking visitors.

Soon enough, more invasive techniques developed. Third-party tracking cookies became an important resource for tracking which ads users clicked. By understanding what kind of advertisements a certain person clicks, the agency can serve ads that users are more likely to click, thus improving CTR. Behind the scenes, the advertising company's program quietly constructs a marketing profile for the user. Given assumptions about a target market, the program can quickly categorize users into age group, gender, and location.

The past decade alone has borne witness to a substantial battle between opposing groups, one attempting to increase tracking and the other advocating for stricter privacy protections. Third party tracking cookies are now blocked by default in almost all contexts, which has eliminated many smaller players from the market. This has led to the rise of the "data brokerage" industry, where first-party data is collected and sold by giants such as Google and Facebook, who together now control 57.6% of the digital ad market.[1]

As the war over internet tracking rages on, the concept of "consent" is far too often overlooked. Users have not even the slightest idea of what kinds of data they automatically provide. "I accept the terms and conditions" is a joke, as practically no one can spare the hours needed to review the terms in explicit detail, and by not accepting the terms the user

[1] http://www.visualcapitalist.com/dominance-google-and-facebook-one-chart/

voids their right to use the service. Browser fingerprinting is one of the most egregious violations of consent, as there is little to no defense against a properly constructed fingerprinting infrastructure.

TO THE COMMUNITY

The following sections describe in detail exactly how effective browser fingerprinting is as a tool for online tracking and deanonymization attacks. Anecdotal evidence exists that suggests browser fingerprinting is seeing widespread use for both digital marketing and fraud protection, but its capacity for online tracking combined with its ease of implementation and high reliability make this a dangerous tool for more maliciously-minded tracking programs.

The only way for standards on fingerprinting to change is for the public to be informed of the severity of this vulnerability. Browser operators such as Google, Microsoft, and Mozilla must do more to defend user privacy.

WHAT IS BROWSER FINGERPRINTING?

Browser fingerprinting was first developed sometime roughly before 2010 as a means of subverting cookie hijacking, where criminals would steal user's cookies from their traffic and use their session data against them.[2] It was a final layer of defense against click fraud and a way of assigning reputation scores to trusted devices. Research is still being conducted into the use of browser fingerprinting as an added layer of session security.[3] However, the technology quickly gained traction in the tracking industry. The capacity to identify individual devices allowed trackers an extremely reliable method of correlating data across

[2] http://blogs.wsj.com/digits/2010/12/01/evercookies-and-fingerprinting-finding-fraudsters-tracking-consumers/
[3] *SHPF: Enhancing HTTP(S) Session Security with Browser Fingerprinting*. Thomas Unger, Martin Mulazzani, Dominik Fruhwirt. 2013. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6657249

websites with individual users at a level of accuracy well beyond dynamically-assigned IP addresses and easily-cleared cookies.

Browser fingerprinting works by analyzing a multitude of device-specific features and combining those metrics to create a fingerprint "hash" for that device.[4] A backend server keeps track of the fingerprint and all actions associated with that fingerprint. A browser fingerprint is unlikely to change significantly over time, as it based on features that are either immutable, hard to disguise, or extremely unlikely to change unless the user is immediately aware of the tracking dangers. See **Table A** for a description of common browser fingerprinting features.

THE VULNERABILITIES OF THE MODERN WEB

The modern web exposes a number of details about the user's device, either directly or discoverable through testing. This is largely a result of a greater need to comply with different device types, support for different features, or simply leaking data by including too much information in protocol requests.

The Panopticlick/EFF study, which will be discussed in the next section, included user agent, HTTP headers, the presence of cookies, screen resolution, timezone, a list of browser plugins, a list of system installed fonts, and a test for planting a supercookie. The list of potential features, however, is enormous. The open-source library fingerprintjs2[5] aggregates 25 different features with varying degrees of entropy, and the authors list an additional 13 categories from which new features could be implemented.

Legacy technology has often caused vulnerability headaches, which has led to the phasing out of many technologies such as Java applets and Adobe Shockwave. One common

[4] https://panopticlick.eff.org/about#browser-fingerprinting
[5] https://github.com/Valve/fingerprintjs2

vulnerability which is quite often unique to individual users is system installed fonts. While JavaScript does not directly permit browsers to get the full list of fonts, if Adobe Flash is supported then it returns all fonts with one simple call. Google recently announced its intention of phasing out Adobe Flash in an upcoming Chrome releasing, citing that “Today, more than 90% of Flash on the web loads behind the scenes to support things like page analytics.”[6] This simple measure will go a long way towards helping minimize vulnerabilities.

Table A: Selected Features Commonly Used in Browser Fingerprinting

Feature	**Definition**
User Agent	Information given by the web browser telling the website about the browser version and operating system
Language	Language the browser is in; used for localization
Screen Resolution	The width and height of the user’s screen
Timezone	The local timezone of the user’s computer
DoNotTrack	Whether the user has indicated they wish to opt-out of tracking
Installed Fonts	The fonts supported in the browser, typically retrieved from Adobe Flash but can also be detected by other means
Canvas Fingerprinting	The program attempts to render an image on the webpage

[6] https://blog.google/products/chrome/flash-and-chrome/

	using the HTML5 canvas. Subtle differences in the rendering of the image can lead to identification
Browser Plugins	A list of plugins installed in the browser
Cookies Enabled	Whether or not the user allows cookies on that page
Benchmark Tests	Evaluate how the browser's JavaScript engine performs against a list of published benchmarks. This information is often used to supplement that of the user agent string

THE FEASIBILITY OF BROWSER FINGERPRINTING

Browser fingerprinting has a number of advantages over traditional means of tracking. In 2010 Peter Eckersley, in association with the Electronic Frontier Foundation (EFF), conducted a study of browser fingerprinting techniques.[7] The experiment entailed collecting fingerprints from visitors of https://panopticlick.eff.org/, who were expected to already have a high awareness of privacy concerns. As such, the results of the experiment are skewed towards those who have already adopted measures of protecting their identity on the internet. This only makes the results even more appalling.

Out of 470,161 browsers sampled, 83.6% of the users had an instantaneously unique fingerprint, and a further 5.3% had an anonymity set of size 2, meaning that there was only one matching fingerprint in the entire set. Having Adobe Flash or JavaScript enabled only worsened the problem, resulting in 94.2% of users having a unique fingerprint with a further

[7] *How Unique is Your Web Browser?* Peter Eckersley. 2010. https://panopticlick.eff.org/static/browser-uniqueness.pdf

4.8% with an anonymity set of size 2. Additionally, despite the fact that the website listed specific techniques and strategies for changing or masking browser fingerprints, of the 8,833 users who returned after a period of more than 24 hours (as measured using tracking cookies), only 37.4% registered a change in their fingerprint.

These statistics paint a grim picture. Even for an audience of privacy-conscious users, who are informed of the dangers of browser fingerprinting, this technique provides an extremely reliable method for tracking individual users across the internet. The authors of the paper additionally provide a simple algorithm for detecting changes in fingerprints. When a fingerprint is detected, they check to see if there is any fingerprint which, aside from one non-matching field, is otherwise the same. If there were multiple potential matches, it would not make any guess. However, it still made a guess in 65.56% of cases, with 99.1% of them correct guesses. The data confirms that browser fingerprinting is an extremely reliable method of tracking.

Each feature used to create the fingerprint provides a little bit of information entropy. The more features collected, the more likely it is that users can be distinguished. The entropy of a feature is a function of its probability. For an even probability distribution, entropy is much higher, as the information given by knowing the outcome is expected to be more informative. On the other hand, an event which has a guaranteed outcome has zero entropy, as knowing the outcome provides no additional information.

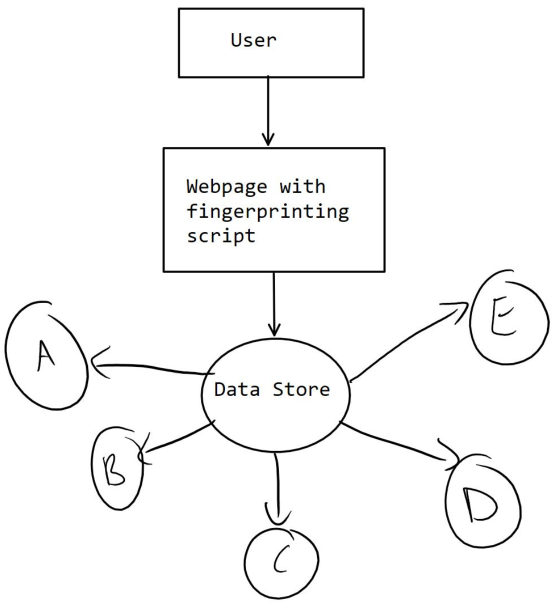
Entropy is a critical measure for fingerprinting techniques. For a single bit of entropy, there are 2^1=2 expected outcomes for a single event, no better than a coin toss. For 18.1 bits of entropy (the measure computed by the EFF study), knowing a single fingerprint, we would only expect one in every 286,777 fingerprints to match.

BROWSER FINGERPRINTING IN THE WILD

Browser fingerprinting, while a powerful tool, is often part of a larger suite of tracking techniques. Fingerprinting operates through the JavaScript engine as part of an analytics program. Any website that runs the fingerprinting analytics script will be able to track users through a shared data store.

The **figure below** describes how a user accessing a website with a fingerprinting script will be compromised across the entire network of websites associated with that analytics program. Upon accessing the website, the user's fingerprint is determined and then pushed to a central data store. When the user accesses any of the websites A-E, they will be recognized, even though they do not have any cookies associated with those websites.

A 2016 study, part of the Princeton University WebTAP Project, found that fingerprinting scripts were still present on 5% of the top 1000 Alexa-ranked websites.[8] They found, additionally, that modern fingerprinting methods were relying more heavily on details that were extremely difficult for a user to modify, such as the audio stack and the display rendering stack. One likely explanation is that companies were not satisfied with the often-changing or easily maskable feature measurements, such as the user agent string, which changes with every browser update, and opted for more rigid assessments.



One critical Flash vulnerability, called "Local Storage Objects," has given rise to what is described as a "zombie cookie" or "supercookie." Supercookies are almost impossible to remove. They work by populating numerous different storage areas with tracking information. Even if the user deletes the cookie from the browser, it can still come back. The other storage locations will repopulate the cookie, so long as it exists in any one of them. Combined with browser fingerprinting, the cookies can track changes in fingerprints while also relying upon the fingerprint as a final line of defense against the most privacy-conscious users. This technique is known as "cookie syncing."[9]

APPLICATIONS

One significant application of browser fingerprinting is the deanonymization of Tor users.[10] Tor, by default, comes with JavaScript enabled. Most of the modern web is nonfunctional with JavaScript disabled, but Tor websites are expected to work without JavaScript. Users who forget to turn off JavaScript may find themselves vulnerable to analytics and tracking methods.

Interestingly, it is possible to use specific fingerprinting techniques to track users in a cross-browser fashion, with a fingerprint unique to their device. Many advanced fingerprinting techniques rely on details about the device and ignore the variable information that comes with the browser or the JavaScript engine. This fingerprint, unique to the device, can be used to identify Tor users in additional contexts.

The most dangerous applications of browser fingerprinting rely on its ability to operate as an immutable third party tracking mechanism. Hypothetically speaking, if all devices were uniquely identifiable over the internet, the entire notion of privacy on the internet would be null and void. The device fingerprint, in combination with information about a person, can be used to compromise their identity. A study of the 1990 census data found that 87.1% of people were uniquely identifiable knowing only their date of birth, zip

[9] https://securehomes.esat.kuleuven.be/~gacar/persistent/
[10] http://jcarlosnorte.com/security/2016/03/06/advanced-tor-browser-fingerprinting.html

code, and gender.[11] Thus, it is very feasible to construct a superset of the data constituting direct links between people and devices given a third source of information (Facebook activity, for example). Knowing the device and personal identity permits fingerprinting agencies to know exactly when a person accesses a participating website. This, in turn, could very easily make up part of an invasive NSA project.

A malicious ISP could also take advantage of this capability as well. In 2015, the Great Firewall of China "accidentally" DDoSed Github.[12] Whenever a Chinese internet user visited a website that contained a Baidu-supplied tracker, the injected code triggered requests to constantly load two GitHub pages. Regardless of the motivation for the attack, the attack demonstrated the ease at which silently injected JavaScript could be part of a greater tracking mechanism. If, instead of overloading GitHub, the injected code analyzed the device fingerprint and sent a record of the browsing history to the Chinese government, there would be unprecedented implications for the decline of internet rights.

Fingerprinting could also provide a cookie-free means of blacklisting certain users. Say your blog was being pestered by a particular commenter. You could request to have them blacklisted, and suddenly their device becomes incapable of accessing any of the sites in your blogging network (harkening back to the days where IP bans were an effective means of blacklisting).

CONCLUSION

The modern web is here to stay, in spite of all the vulnerabilities and frightening privacy implications that entails. Online tracking and analytics-based marketing are expected to continue growing so long as the free services that come with them are good enough.

[11] http://dataprivacylab.org/projects/identifiability/paper1.pdf
[12] http://arstechnica.com/security/2015/04/ddos-attacks-that-crippled-github-linked-to-great-firewall-of-china/

Privacy is, frankly, not a real concern for most web users. The danger of fingerprinting is that it goes beyond traditional means by establishing a unique, unavoidable method of identification. Fingerprinting is not only simple and easy to implement but also extremely effective at doing its job.

On the positive side, the previously mentioned Princeton study also reported that, from 2014 to 2016, there was a decrease in the number of webpages running fingerprint analytics scripts, and traced the cause to the public backlash surrounding their initial report in 2014. While there is little the general populus can do to stop individual actors from such activities, public pressure is important enough to prevent the adoption of unethical models by the big publicly-facing companies. So long as Google and Facebook dominate the tracking market, we may yet hold out hope that those institutions have enough respect for privacy advocates to avoid expanding tracking too far. But with the efficiency and effectiveness of fingerprinting methods, perhaps it is only a matter of time.

KEY REFERENCES

How Unique is Your Web Browser? Peter Eckersley. 2010.

https://panopticlick.eff.org/static/browser-uniqueness.pdf

SHPF: Enhancing HTTP(S) Session Security with Browser Fingerprinting. Thomas Unger, Martin Mulazzani, Dominik Fruhwirt. 2013.

http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6657249

https://github.com/Valve/fingerprintjs2

User Tracking on the Web via Cross-Browser Fingerprinting. Károly Boda, Ádám Máté Földes, Gábor György Gulyás, Sándor Imre. 2011.
http://link.springer.com/chapter/10.1007/978-3-642-29615-4_4

Online Tracking: A 1-million-site Measurement and Analysis. Steven Englehardt, Arvind Narayanan.
http://randomwalker.info/publications/OpenWPM_1_million_site_tracking_measurement.pdf

https://webtransparency.cs.princeton.edu/webcensus/

https://securehomes.esat.kuleuven.be/~gacar/persistent/