Poster: Detection and Prevention of Web-based Device Fingerprinting

Daehyeok Kim
Researcher
Cyber Security Research Center
KAIST
Email: dhkim7@kaist.ac.kr

I. Motivation

Web tracking is a set of technologies that allows websites to create profiles of their visitors. While a website owner might utilize such profile to provide its users with personalized advertisements or anti-fraud feature, tracking of users is generally considered a problem that brings user privacy under attack.

According to a recent survey by Mayer et al. [1], web tracking technologies can be roughly divided into two groups: stateful and stateless. In stateful tracking, a tracking company utilizes stateful information that can be gathered from browser cookies, Flash cookies, ETag cookies, and HTML5 local storage. Evercookie [2] provides a reference implementation for many stateful tracking techniques. On the other hand, stateless tracking, also called device fingerprinting, captures the properties of browser elements through JavaScript, Flash, or other plugins and forms a nearly unique identifier. In [3], Eckersley shows that user's device can be uniquely identified with stateless properties including user-agent, time zone, screen resolution, fonts installed, plugins installed, and cookies enabled. Also, recent studies [4], [5] show that stateless tracking methods are used by various tracking companies in the wild.

Several groups of researcher have considered countermeasures against stateful web tracking and policy makers in U.S. and European Union have attempted to regulate it. Also, most modern web browsers supports the opt-out feature for stateful web tracking. For example, a web browser can disallow third-party websites from utilizing cookie information and the Do Not Track (DNT) HTTP header field allows users to signal their tracking preferences to websites.

However, these countermeasures are not suitable for stateless web tracking since they mostly focus on stateful information such as browser cookies. Among the fingerprinting information captured for uniquely identifying a device, it has been shown that the list of installed fonts and plugins provide relatively more unique values [3]. In order to fingerprint the list of font from the web browser, tracking scripts have to utilize a combination of properties of HTML elements such as *fontFamily*, *offsetHeight*, and *offsetWidth*. Unfortunately, since these properties are widely used for both tracking and non-tracking scripts, it is difficult to distinguish a fingerprinting JavaScript code from a normal script code.

Several countermeasures against device fingerprinting have been proposed. Some web browser extensions and plugins would randomize the value of user-agent, screen resolution, or properties of HTML element when their values are retrieved. However, these countermeasures may not be effective because they often cause breakage of the rendered web page and the existence of such extensions can be another kind of fingerprint. In addition, although users can tell their tracking preferences to websites through the DNT header, it has been shown that DNT preferences are usually ignored by web trackers [5].

In this study, we present FPBlock, a system that detects web-based device fingerprinting and prevents users from being fingerprinted. FPBlock takes a different approach from those of existing countermeasures, which randomize the value of properties or rely on a blacklist and the DNT header. FPBlock detects fingerprinting scripts based on a dynamic analysis of JavaScript codes embedded in the websites; it then prevents those codes from leaking the user's fingerprint to the third-party server. Since FPBlock automatically detects fingerprinting functionalities included in any JavaScript and uses them as features, we believe that it provides a more practical, effective and robust method than those of existing approaches. In this poster, we focus on detecting JavaScript based fingerprinters; we are currently exploring countermeasures against other types of fingerprinters.

II. Design

Our goal is to prevent third-party fingerprinting scripts from leaking the user's fingerprint information. Our approach is to analyze the JavaScript codes embedded in websites and determine whether those codes performs any device fingerprinting behaviors such as enumerating the fonts or plugins installed. We assume that the enumerating fonts or plugins installed in the user's system are unusual behaviors of websites. That is, normal websites have no clear reason to retrieve such information from users when rendering their web pages correctly. If such fingerprinting behavior is detected, all attempts to transmit data to the corresponding third-party tracking server will be blocked.

We studied the characteristics of device fingerprinting JavaScript codes including the following:

- **Origin domain of the script:** Fingerprinting scripts are usually loaded from third-party tracking servers, which are mostly located at domains different from those of the host websites.
- **Capability of font enumeration:** Existence of font enumerating capability in a script indicates the high possibility of that script being a fingerprinting script. Unfortunately, it is not a trivial task to determine whether a script has font

enumeration functionality. Below, we describe more details on how FPBlock detects font enumeration functionality.

- **Capability of plugin enumeration:** Existence of plugin enumeration capability in the script is another strong indicator of a fingerprinting script. Detection of the behavior of plugin enumeration can be captured by tracking the access to the *window.navigator.plugins* property.
- **Interaction with a remote server:** A fingerprinting script usually interacts with a third-party remote server to send the user's fingerprinting data.

Note that the above list describes only a part of all characteristics we found. We plan to include more details and other interesting features.

Since there is no explicit JavaScript API to retrieve a list of installed fonts, a script developer has to find alternative ways. We found that any JavaScript that enumerates installed fonts has certain characteristics. It changes the *fontFamily* property to render a string with different fonts and retrieves *offsetWidth* and *offsetHeight* to measure the size of rendered character string. With this observation, FPBlock checks whether these properties of identical HTML elements(e.g., SPAN or DIV) are regularly altered and uses such alteration as a key indicator of existence of font enumeration functionality.

When a user visits a website, FPBlock tracks all JavaScript and DOM activities during the execution of JavaScript codes; it also tracks their origin domains. The core of FPBlock is that it can analyze JavaScript codes and decide whether those codes contain routines that enumerate installed fonts or plugins when a webpage is being loaded. For example, to detect font enumeration, each time the JavaScript code accesses properties such as *fontFamily*, *offsetWidth*, and *offsetHeight*, FPBlock tracks the corresponding HTML elements and properties being accessed and maintains the information in a sequential manner. When the JavaScript code makes an attempt to send the such data using *XMLHttpRequest* or other possible methods, the analyzer, based on recorded behavior, makes a decision as to whether the code contains any fingerprinting functionality. When FPBlock detects such functionality, it blocks the data transmission to the remote server.

We believe that FPBlock is more effective than any of the blacklist based approaches because it detects the behavior of fingerprinting in runtime rather than relying on a list of third-party domains, which could be static. Also, compared to randomization approaches, FPBlock is more efficient and reliable because it affects the behavior of JavaScript(e.g., blocking the data transmission) only if it detects fingerprinting functionality.

III. Open questions

In this section, we describe open challenges that will need to be studied in detail to improve the performance of FPBlock.

Handling other types of fingerprinting techniques: Although we focused on the case of JavaScript, Flash with ActionScript can also be used for fingerprinting. In contrast to JavaScript, Flash uses its own proprietary interpreter(i.e., Flash player), which is not easy to handle it at the web browser level.

Exploring additional features of device fingerprinters: So far, we have utilized the basic characteristics of JavaScript that attempts to enumerate installed fonts and plugins. There might exist uncovered features which can further improve the effectiveness of the system.

Defending against passive fingerprinting: Unlike active fingerprinting through scripts or plugins, there are other methods of implicit device fingerprinting. Such techniques are called passive fingerprinting; the information they collect includes the user's IP address, Operating system, and HTTP accept headers. Although these fingerprints may contain less unique information compared to that of active fingerprints, they can be complementary to active ones. More studies are needed to defend against these types of passive fingerprinting.

IV. Implementation and Preliminary result

We implemented the prototype of FPBlock on a web browser emulator that was built based on SpiderMonkey [6] and env.js [7]. We evaluated the prototype with the device fingerprinting JavaScript samples collected by [5].

Our preliminary results show that FPBlock successfully detects all 8 fingerprinting JavaScripts in runtime and is able to block data transmission to third-party tracking servers. We are currently performing extensive studies on various websites and will provide the detailed experiment result.

V. Conclusion

In this study, we propose FPBlock to detect web-based device fingerprinting JavaScript components and prevent them from leaking user fingerprints to third-party servers. We studied the characteristics of JavaScript codes that fingerprints a device; we also show that JavaScript dynamic analysis can be an effective way to detect and control such fingerprinters. As an ongoing work, we are currently implementing FPBlock on open source web browser and plan to perform an extensive evaluation.

References

[1] J. Mayer and J. Mitchell, "Third-party web tracking: Policy and technology," in *Security and Privacy (SP), 2012 IEEE Symposium on*, May 2012, pp. 413–427.

[2] evercookie - virtually irrevocable persistent cookies. [Online]. Available: http://samy.pl/evercookie/

[3] P. Eckersley, "How unique is your web browser?" in *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, ser. PETS'10, 2010, pp. 1–18.

[4] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting," in *Security and Privacy (SP), 2013 IEEE Symposium on*, May 2013, pp. 541–555.

[5] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel, "Fpdetective: Dusting the web for fingerprinters," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13, 2013, pp. 1129–1140.

[6] Spidermonkey - mozilla. [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey

[7] Env.js - bringing the browser. [Online]. Available: http://www.envjs.com/