Towards accurate detection of obfuscated web tracking

Hoan Le, Federico Fallace and Pere Barlet-Ros
Universitat Politecnica de Catalunya, UPC-BarcelonaTech
Barcelona, Spain
{hoan, fallace, pbarlet}@ac.upc.edu

***Abstract*—Web tracking is currently recognized as one of the most important privacy threats on the Internet. Over the last years, many methodologies have been developed to uncover web trackers. Most of them are based on static code analysis and the use of predefined blacklists. However, our main hypothesis is that web tracking has started to use obfuscated programming, a transformation of code that renders previous detection methodologies ineffective and easy to evade. In this paper, we propose a new methodology based on dynamic code analysis that monitors the actual JavaScript calls made by the browser and compares them to the original source code of the website in order to detect obfuscated tracking. The main advantage of this approach is that detection cannot be evaded by code obfuscation. We applied this methodology to detect the use of *canvas-font tracking* and *canvas fingerprinting* on the top-10K most visited websites according to Alexa's ranking. Canvas-based tracking is a fingerprinting method based on JavaScript that uses the HTML5 canvas element to uniquely identify a user. Our results show that 10.44% of the top-10K websites use canvas-based tracking (canvas-font and canvas fingerprinting), while obfuscation was used in 2.25% of them. These results confirm our initial hypothesis that obfuscated programming in web tracking is already in use. Finally, we argue that canvas-based tracking can be more present in secondary pages than in the home page of websites.**

***Index Terms*—Fingerprinting; Canvas fingerprinting; Canvas-based Tracking; Obfuscated tracking; Web tracking; JavaScript**

I. Introduction

In the last years, several works have shown that many services on the Internet systematically collect and use a large amount of users' personal information [1]. This phenomenon is called *web tracking* and it is a fast-growing business that counts hundreds of companies. The prevalence and pervasiveness of web tracking is considered as one of the most serious problems on the current Internet [1]–[7].

The data collected by web trackers include information of technical nature, such as the IP address or the browser in use, but also more sensitive information, such as the list of visited web sites or our favorite products and interests. Some previous studies have demonstrated that this personal information is used for many different purposes, including targeted advertising [8], [9], price discrimination [10], [11], or even the assessment of our health [12], [13] and financial condition [14]–[16].

Another relevant aspect is that a large amount of web tracking methods are not deployed directly by first-parties, but by third-party trackers present on the visited websites that load content from other domains, such as advertisers and analytics companies. A recent work [6] showed that around 46% of the websites in the top-10K Alexa ranking contains at least one third-party tracker. Third-party trackers are a serious privacy threat, since they can collect information in the background from many different websites and combine them without explicit knowledge of the user.

Over the last few years, web tracking technologies rapidly evolved from simple methods to more complex techniques that try to overcome the privacy protection mechanisms provided by modern browsers. Several tracking services also share user identifiers, so there is a possible adoption of the Cookie Matching practice that links and unifies user identifiers from different trackers [17]. In addition, trackers have combined JavaScript with other web technologies (e.g., Flash or Java), making it more difficult to avoid being tracked.

In our recent survey [1], we reviewed the various techniques currently used by web services in order to track users. They were classified into different groups, depending on whether they are based on session identifiers, client storage, client caches, user fingerprinting or a combination of them. In this paper, we focus on user fingerprinting techniques and, in particular, on the accurate detection of canvas-based fingerprinting methods. Canvas fingerprinting is currently the most popular and widespread fingerprinting method on the Internet [2], [5].

The main idea behind canvas-based tracking is to add an HTML5 canvas element with particular fonts, text or shape in the website and to take the user rendered value as a fingerprint, due to the fact that the rendered value heavily depends on the used hardware and software installed [18]. The strength of this technique is that it does not depend on any client-based storage, so it is almost impossible to avoid, even when using the private browsing mode or anonymous networks (e.g., Tor, VPN or I2P). Canvas fingerprinting is based on JavaScript, so the source code has to be delivered to the user's browser in order to be executed. A simple approach to detect JavaScript-based fingerprinting is by static code analysis on the client side, looking for suspicious calls that are usually related to tracking. However, this approach is very easy to evade by using code obfuscation tools, which render code inspection ineffective. Code obfuscation is a common method used to hide and protect JavaScript code from its theft and reuse by other developers. A myriad of tools to obfuscate JavaScript code are easily available (e.g., [19]–[22]).

Our research hypothesis is that web tracking is becoming obfuscated. We refer to obfuscated tracking as a tracking code that has been modified in a way it cannot be explicitly understood by a human nor easily detected by pattern-matching

methods. Thus, this kind of tracking is difficult to detect by static analysis of the original source code.

In this paper, we propose a new methodology based on dynamic code analysis in order to uncover obfuscated web tracking. In particular, we monitor the actual JavaScript calls made by the browser and compare them with the original JavaScript code in order to detect obfuscated canvas-based tracking, including canvas fingerprinting and canvas-font fingerprinting. The main advantage of this approach is that the detection of web tracking cannot be evaded by using code obfuscation techniques.

The main contributions of our paper are as follows:

- We propose a novel methodology based on dynamic code analysis that tracks the actual calls made by the JavaScript API and compares them to the original HTML/JavaScript code to uncover obfuscated web tracking.
- We apply our methodology to the top-10K websites according to Alexa's ranking as well as on the main links of the top-100 Alexa ranking websites [23].
- We show that at least 9.13% of websites use canvas fingerprinting on the top-10K home pages, while 2.2% obfuscate it. In addition, 1.65% of the analyzed websites use canvas-font fingerprinting, from which 0.05% is obfuscated.
- Overall, 10.44% of the analyzed websites use canvas-based tracking, while obfuscation is detected on 2.25% of them. Some of the sites were using both techniques.
- We proved our initial hypothesis that web tracking is becoming obfuscated and proposed a promising direction to uncover it. To the best of our knowledge, this is the first work to study obfuscated web tracking and how to uncover it.

The rest of the paper is organized as follows. Section II presents the basic background on canvas fingerprinting and obfuscated web tracking. Section III describes the methodology we used. Our results and discussion on uncovering obfuscated canvas fingerprinting are presented in Section IV. Section V reviews the state-of-the-art in web fingerprinting. Finally, we present our conclusions and future work in Section VI.

II. BACKGROUND

In this section, we present an introduction to canvas-based fingerprinting and obfuscated web tracking.

A. Canvas fingerprinting

In the last years, the web browser has become the main tool to access the content available on the Internet. In order to provide a correct visualization of the online content, the browser has to provide some information about the installed software and the used hardware to web services, which is needed in order to render contents or to serve device compatible media. The problem is that the APIs commonly used for the correct visualization are flexible enough to be used to obtain a fingerprint, which is unique for the device. This practice is called web-based device fingerprinting and it has important privacy and security implications.

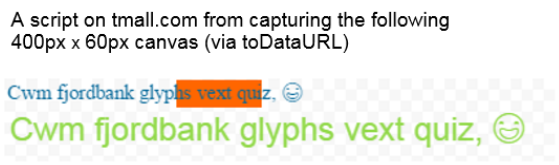

Fig. 1. Example of the output of canvas fingerprinting on tmall.com

We can consider the Eckersley's experiment in 2010 [4] as the official discovery of the fingerprint. He argued that the information provided by the browser, such as screen dimensions or installed fonts, could be combined to create a device-specific fingerprint. Different attributes were used with different priorities depending on how common they are among users and how stable they are in a device. The results of the experiment were that 94.2% of the devices had a unique fingerprint. These results are limited to devices using JavaScript and Flash, but they are still worrisome if we think that can be used to identify and track a user without stateful client-side technologies. As a result, the trackers are able to identify users also if they avoid the use of browser or Flash cookies, circumventing users' preferences about tracking and limitations imposed on cookies by Europe and United States regulations.

Web tracking companies are developing more complex and advanced mechanisms, where canvas fingerprinting is one of the last to be discovered. This type of fingerprinting uses the HTML5 element <Canvas> that provides a drawable screen area. The functioning is simple. The script lets the browser draw a text or image in the canvas area and then reads the rendered image back. The value will be different depending on the installed software and used hardware and it can be used to uniquely identify a user. Canvas fingerprinting uses mainly four JavaScript methods: `fillText()` and `strokeText()` are used to draw text with a given font, size and background color. Then, `toDataURL()` or `getImageData()` are used to send back a Base64 encoding of the PNG image that contains the whole content of the canvas element [2], [18]. Other methods, such as `MozFetchAsStream()` or `ExtractData()`, can be used for this purpose, but they were not investigated in this paper. An example of the output of canvas fingerprinting is given in Fig. 1, while a sample of a canvas fingerprinting code is provided in Fig. 2.

Canvas fingerprinting brings Internet tracking to an entirely new level of invasiveness. It is performed without the user's prior knowledge or consent, hence it is very difficult to know when and where tracking happens. Moreover, it does not need client-based storage, so removing it to create a new clean profile is almost impossible.

B. Canvas-based font fingerprinting

Since the born of fingerprinting in [4], it was clear that the list of the installed fonts was very useful to identify a specific

device. Although currently browsers do not provide so easily this list, other alternative ways can be used to obtain it.

One method is to use Flash and, in particular, its scripting language ActionScript that provides APIs and methods to obtain this list. When Flash is not available, it is still possible to use JavaScript. Some functions are able to retrieve the dimension in pixels of a text, and consequently to understand if a specific font is installed or not in the analyzed device. Indeed browsers usually render a text with the default "sans" if the given font is not installed, so the size of "sans" for a random sentence is taken and used as sample [24].

The specific function for canvas is `measureText()`. Clearly, it should be used to correctly visualize the text, but if the number of the calls is higher than normal, we can conclude that it is used for fingerprinting. In [5] this number was set to 50, so scripts with less than 50 calls of `measureText()` in the same text string were filtered out as false positives. We followed also this criterion in our analysis.

C. Code obfuscation

Most modern tracking methods are based on JavaScript. Therefore, a simple way to avoid detection of tracking code when using static code analysis tools is the use of code obfuscation.

Code obfuscation is the process of modifying the source code, to avoid or delay the understanding of a program, in a way it is very difficult to read, modify and reuse by other programmers or competitors [25]. Indeed it can improve security, prevent fraud and protect intellectual property; it is a useful tool against reverse engineering attacks based on static analysis and limits the impact of dynamic analysis.

A common misconception is that obfuscation is a kind of encryption, but the former is still executable without any decryption or deobfuscation, so the concepts are different. Obfuscation could also be confused with minification, but the second one has the only purpose of reducing the code size to make the program smaller and faster, although some techniques are equal or similar (removing unnecessary spaces and new lines for example). When the source code has to be distributed to be used, for instance by web services using JavaScript, it can be a really powerful method to protect programmers' work. On the other hand, in the web tracking context it could be easily used to hide tracking code and avoid its uncovering and blocking (Internet is full of tools to obfuscate code [19]–[22]).

III. METHODOLOGY

In this section, we describe our methodology to uncover obfuscated canvas-based tracking using dynamic code analysis. As stated above, it is not possible to detect obfuscated tracking code by static analysis of the Javascript source code. Fig. 2 illustrates this problem with two examples that, using Javascript, extract a canvas fingerprint of the user. The first file contains a plain-text fingerprinting script that we developed, while the second one contains an obfuscated version that we created using a free web service [22]. The fingerprints obtained by the two scripts are exactly the same, but static analysis tools are not able to detect the fingerprinting code in the obfuscated version, as the original Javascript calls are not visible.

Our method uncovers obfuscated fingerprinting by comparing the original JavaScript source code with the actual calls made to the JavaScript engine of the browser. The actual API calls are impossible to hide, since these functions are executed directly by a JavaScript interpreter inside the user browser. Intuitively, the API calls for obfuscated and non-obfuscated tracking are exactly the same, although obfuscated tracking code is modified in a way that cannot be easily recognized by a human when looking at the source code or using pattern matching methods.

In particular, we modified the source code of the Firefox browser to log the API calls to the Javascript engine. All the HTTP(s) communications are intercepted and collected in a file using mitmdump (a SSL-capable HTTP proxy), as shown in Fig. 3. The analysis of the mitmdump logs and the intercepted API calls allow us to uncover obfuscated tracking as follows:

1) If a particular API function is called (in the browser) but it is not present in the original source code (in the mitmdump log), it means that the JavaScript code is obfuscated.
2) If the API function is present both in the mitmdump log and in the log of the API calls made by the browser, it means that the JavaScript is not obfuscated.
3) If a call to an API function is present in the source code, but the function does not appear in the log of the API calls, it means that the fingerprinting code was present but never executed.

In any case, when an API function was called, we can figure out when it was exactly called (after which user's action) by looking at the execution log.

We used Selenium WebDriver to automatically perform the crawling of a list of websites, and also to scroll down and up to let the page show all dynamic contents. We considered those API calls that are typically used for tracking purposes and that were already employed in several previous works based on static code analysis [1]. We analyzed and crawled the most popular top-10K websites according to Alexa's ranking.

By collecting and analyzing most invoked JavaScript calls, we uncover websites using obfuscated canvas tracking. In a second experiment, we also scraped around 3000 links present in the home pages of the top-100 Alexa websites for a deeper analysis of canvas and obfuscated tracking. In particular, we hypothesize that canvas tracking methods and obfuscation could be more present in secondary pages different than the home page for two reasons. The first is that useful information about our interests, our searched objects and so on, are more likely to be exposed in secondary pages than on the home pages. The second is that most of the previous works that analyzed the presence of web tracking (e.g., [2], [5]), focused only in the home pages, so the presence of web tracking on 2nd or 3rd level domain links is still unknown and could also

Plain-text version

Your fingerprint is:
**
CWCAYAAABkW7XSAAAZY0lEQVR4Xu2de3iU1Z3Hv...**

```
var canvas = document.createElement("canvas");
var ctx = canvas.getContext("2d");
// https://www.browserleaks.com/canvas#how-does-it-work
var txt = "Cwm fjordbank glyphs vext quiz, https://github.com/valve á1½ ";
ctx.textBaseline = "top";
ctx.font = "70px 'Arial'";
ctx.textBaseline = "alphabetic";
ctx.fillStyle = "#f60";
ctx.fillRect(125, 1, 62, 20);
ctx.fillStyle = "#069";
ctx.fillText(txt, 2, 15);
ctx.fillStyle = "rgba(102, 204, 0, 0.7)";
ctx.fillText(txt, 4, 17);
return canvas.toDataURL();
```

Obfuscated version

Your fingerprint is:
...

```
var _0x1f5a=["\x63\x61\x6E\x76\x61\x73",
"\x63\x72\x65\x61\x74\x65\x45\x6C\x65\x6D\x65\x6E\x74",
"\x32\x64",
"\x67\x65\x74\x43\x6F\x6E\x74\x65\x78\x74",
"\x43\x77\x6D\x20\x66\x6A\x6F\x72\x64\x62\x61\x6E\x6B\x20\x67\x6C\x79\x70\x68\x73\x20\x76\x65\x78\x74\x20\x71\x75\x69\x7A\x2C\x20\x68\x74\x74\x70\x73\x3A\x2F\x2F\x67\x69\x74\x68\x75\x62\x2E\x63\x6F\x6D\x2F\x76\x61\x6C\x76\x65\x20\u1F60",
"\x74\x65\x78\x74\x42\x61\x73\x65\x6C\x69\x6E\x65",
"\x74\x6F\x70",
"\x66\x6F\x6E\x74",
"\x37\x30\x70\x78\x20\x27\x41\x72\x69\x61\x6C\x27",
"\x61\x6C\x70\x68\x61\x62\x65\x74\x69\x63",
"\x66\x69\x6C\x6C\x53\x74\x79\x6C\x65",
"\x23\x66\x36\x30",
"\x66\x69\x6C\x6C\x52\x65\x63\x74",
"\x23\x30\x36\x39",
"\x66\x69\x6C\x6C\x54\x65\x78\x74",
"\x72\x67\x62\x61\x28\x31\x30\x32\x2C\x20\x32\x30\x34\x2C\x20\x30\x2C\x20\x30\x2E\x37\x29",
"\x74\x6F\x44\x61\x74\x61\x55\x52\x4C"];
var a1=document[_0x1f5a[1]](_0x1f5a[0]);
var ctx=a1[_0x1f5a[3]](_0x1f5a[2]);
var txt=_0x1f5a[4];
ctx[_0x1f5a[5]]=_0x1f5a[6];
ctx[_0x1f5a[7]]=_0x1f5a[8];
ctx[_0x1f5a[5]]=_0x1f5a[9];
ctx[_0x1f5a[10]]=_0x1f5a[11];
ctx[_0x1f5a[12]](125,1,62,20);
ctx[_0x1f5a[10]]=_0x1f5a[13];
ctx[_0x1f5a[14]](txt,2,15);
ctx[_0x1f5a[10]]=_0x1f5a[15];
ctx[_0x1f5a[14]](txt,4,17);
return a1[_0x1f5a[16]]();
```

Fig. 2. Example of canvas fingerprinting script and its obfuscated version

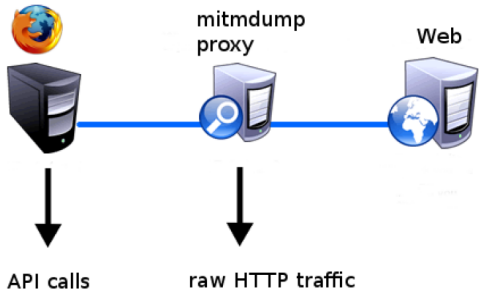

Fig. 3. Main components of our framework

be moved as a consequence of the results made public from these previous works.

IV. EXPERIMENTS AND DISCUSSION

In this section, we analyze the presence of canvas-based tracking on the Internet using the methodology described in Sec. III. Firstly, we automatically crawled the top-10K home pages according to Alexa and detected which of them make calls related to canvas-based tracking (e.g., `fillText()`, `strokeText()`, `toDataURL()`, `getImageData()`). Then, we analyzed this dataset to uncover the use obfuscated programming and to validate our hypothesis on the use of obfuscated web tracking. Finally, we compared the tracking ratios observed in the home pages with those found in the other links present in the website using a smaller sample (top-100 from Alexa's ranking), given that the number of links to be analyzed increases significantly when going deeper into the website hierarchy. Our first results indicate that web tracking can be more present on the secondary pages than in the home pages, but we leave the confirmation with larger datasets as future work. Our experiments were performed in April 2016.

A. Canvas-based fingerprinting on the top-10K home pages

In Fig. 4 we observe that 9.13% of the analyzed websites were using canvas fingerprinting (Canvas FP); 6.93% used plain-text code (the functions were also present in the source code), while 2.2% used obfuscated canvas tracking (they made JavaScript calls in the browser, but the methods were not present in the original source code). We also observed canvas-font tracking, but the percentages were relatively low; 1.65% of the websites used plain-text canvas-font tracking, while only 0.05% used obfuscation. Overall, at least 10.44% of the analyzed websites used canvas-based tracking on their home pages, while 2.25% of which used obfuscated canvas-based tracking. Note that some of the websites were using both canvas-font and canvas fingerprinting. These results show that more than 2% of the websites that use canvas-based fingerprinting could not be detected by previous methods based on static code analysis or pattern matching. In Table I, we can also observe that the presence of canvas fingerprinting in the first top-100, 1K and 10K websites is different, while the percentage of obfuscation increases for less visited websites.

In order to filter out false positives (i.e., legit usage of the HTML5 canvas element) we implemented similar rules to those proposed in [5]. In particular, we did not consider

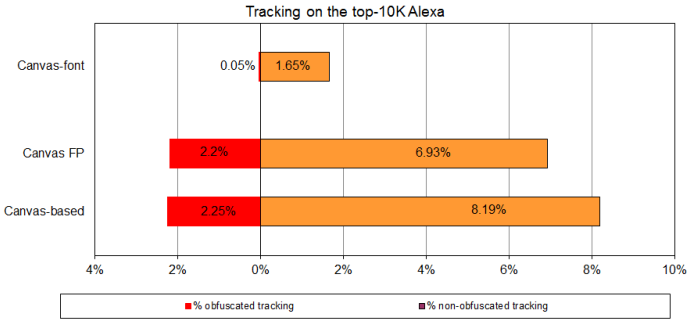

Fig. 4. Canvas-based fingerprinting on the top-10K Alexa websites

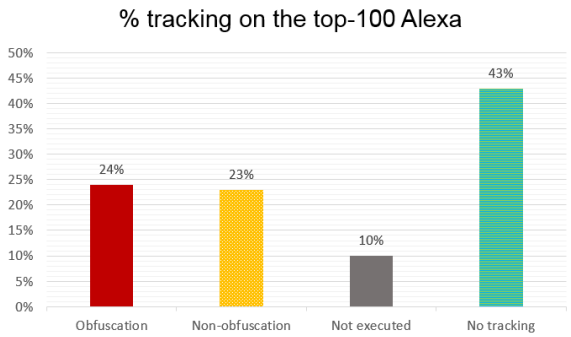

Fig. 5. Canvas-based tracking on the top-100 Alexa websites

TABLE I
CANVAS FINGERPRINTING RATIOS IN THE TOP-100, 1K AND 10K HOME PAGES

Tracking	Top-100	Top-1K	Top-10K
Non-obfuscated	7%	5%	6.93%
Obfuscated	1%	1%	2.2%

TABLE II
CANVAS FINGERPRINTING RATIOS IN THE TOP-100

	Canvas tracking	Obfuscated canvas tracking
Top-100 home pages only	8/100	1/100
Top-100 websites	47/100	24/100

canvas elements with properties of height or width larger than 16 pixels because they are less likely to be used for tracking purposes. Note that the default canvas size is 300x150px. In the case of canvas-font fingerprinting, we also discarded scripts with less than 50 calls to `measureText()` as described in Section II.

Although [5] showed a decrease of the use of canvas fingerprinting compared to previous results reported in 2014 [2], according to some of our experiments with second level pages (see Table II), we observed that part of canvas fingerprinting could have moved from the home pages to inner pages in the website. Although our study may not be directly comparable to these previous works due to the differences in methodology, results also differ because we also consider obfuscated tracking code. As a consequence, our results show a larger percentage of websites that use canvas fingerprinting. In addition, the tests were made in different moments of time. It is also worth noting that some websites, especially Chinese ones, are difficult to reach and need more minutes to be completely loaded. The timeout used when crawling a website in our study is 200 seconds, while in [5] it was 90 seconds. As consequence, the ratio of websites that could not be loaded in our study is 1.2% instead of 8.2%.

B. Canvas-based fingerprinting on secondary pages

According to [5], canvas-based tracking was present on the home page of 3 websites in the top-100. In our analysis, we found 8 websites in the top-100 that use canvas fingerprinting to track users on the home page, including obfuscated and non-obfuscated versions (see Table III). Although [5] only analyzed the use of canvas fingerprinting in the home pages, fingerprinting code could also be present in pages deeper in the website hierarchy. For this reason, in April 2016, we performed a small experiment on the top-100 most visited websites, crawling over 3000 links, including those present in the home pages of the visited websites as well as some important third-level domains links. In Table II and Fig. 5, we show the results of this experiment. When analyzing other links and not only the home pages, we found 47 websites in the top-100 using canvas fingerprinting (39 of them were not in the home page); 24 of these 47 websites were obfuscated, while 23 were in plain-text.

Although our tests were not systematic and did not cover all the links present on the home pages because the number of links quickly explodes, our results seem to indicate that canvas fingerprinting is more present in the secondary pages. We leave the confirmation of this finding with the top-10K Alexa's websites as future work.

V. RELATED WORK

Several works have recently studied the web tracking problem from different perspectives [1]–[7]. These previous studies showed that a large number of websites collect personal information from the users when browsing the web. In our recent

TABLE III
WEBSITES IN THE TOP-100 USING CANVAS FINGERPRINTING IN THE HOME PAGE

Canvas/ Obfuscation	Name websites	Canvas by OpenWPM
Yes/No	**taobao.com**	**Yes**
Yes/No	qq.com	No
Yes/No	sina.com.cn	No
Yes/No	hao123.com	No
Yes/No	sohu.com	No
Yes/No	**tmall.com**	**Yes**
Yes/Yes	***microsoft.com***	***No***
Yes/No	espn.com	No

survey [1], we reviewed the existing tracking mechanisms, their implications and some possible defenses.

A particularly interesting framework to identify and analyze fingerprinting is FPDetective [3], proposed in 2013. They analyzed the presence of tracking on the home pages, using JavaScript-based font probing. The findings suggested that tracking is more widespread than previously thought.

In a remarkable paper [2], Acar et al. analyzed the popularity of canvas fingerprinting on the Internet. They found that canvas fingerprinting is the most commonly used fingerprinting method, present in more than 5.5% of the top-100K Alexa websites. The study discovered a total of 20 canvas fingerprinting provider domains, active on 5542 of the top-100K websites.

S. Englehardt and A. Narayanan in 2016 [5] presented the largest and most detailed measurement of online tracking on the top 1 million websites. They analyzed several tracking techniques, including canvas fingerprinting, canvas-font fingerprinting, WebRTC-based fingerprinting, AudioContext fingerprinting and Battery API fingerprinting. The authors found out that 1.6% of the top-1M websites use canvas fingerprinting. On the top-10K home pages, they found 403 websites (4.03%) using canvas fingerprinting. The measurement was made with OpenWPM, an open source software, already used as the basis of several studies on web privacy and security.

The main difference between our study and these previous works is that we analyze and report the use of obfuscated tracking. To the best of our knowledge, this is the first work to consider the use of code obfuscation in web tracking.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we presented the first results of our investigation on the use of obfuscated web tracking. We proposed a new methodology to uncover the use of obfuscated canvas-based tracking and to efficiently detected canvas fingerprinting on the Internet. Our results confirm the existence of obfuscated canvas-based fingerprinting in the top-10K most visited websites. We also showed that tracking methods can be more present in secondary pages than in home pages.

We conducted a measurement study to determine how often obfuscated tracking is used in the most visited websites. The results show that at least 10.44% of the analyzed websites use canvas-based fingerprinting and 2.25% of them obfuscate it.

As part of future work, we plan to crawl the secondary pages of the top-10K Alexa's ranking to further investigate the problem of obfuscated canvas fingerprinting. In addition, we are currently working on a machine learning solution to automatically detect obfuscated web tracking from passive network measurements without the need of instrumenting the code of the user browser.

VII. ACKNOWLEDGEMENTS

This work was funded by the Spanish Ministry of Economy and Competitiveness and EU FEDER under grant TEC2014-59583-C2-2-R (SUNSET project) and by the Catalan Government (ref. 2014SGR-1427). Hoan Le was supported by an Erasmus Mundus AREAS+ Mobility scholarship, while Federico Fallace was supported by the project ERASMUS+/PROGRAMME COUNTRIES to stay one year at UPC. Authors would also like to thank Marco Mellia and Stefano Traverso from Politecnico di Torino for very useful discussions on web tracking.

REFERENCES

[1] Tomasz Bujlow, Valentin Carela, Josep Sole-Pareta, and Pere Barlet-Ros. A survey on web tracking: Mechanisms, implications, and defenses. *Proceedings of the IEEE*, 105(8), 2017.

[2] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 674–689, New York, NY, USA, 2014. ACM.

[3] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gurses, Frank Piessens, and Bart Preneel. Fpdetective: Dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, CCS '13, pages 1129–1140, New York, NY, USA, 2013. ACM.

[4] Peter Eckersley. How unique is your web browser? In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, PETS'10, pages 1–18, Berlin, Heidelberg, 2010. Springer-Verlag.

[5] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1388–1401, New York, NY, USA, 2016. ACM.

[6] Tai-Ching Li, Huy Hang, Michalis Faloutsos, and Petros Efstathopoulos. *TrackAdvisor: Taking Back Browsing Privacy from Third-Party Trackers*, pages 277–289. Springer International Publishing, Cham, 2015.

[7] Nick Nikiforakis. Web fingerprinting: Who, how and why? In *Proceedings of the OWASP AppSec Research*, Hamburg, Germany, 2013.

[8] Lisa Arthur. Finally! new capabilities for more accurate targeting of facebook ads. *Forbes, http://www.forbes.com*, 2015.

[9] Microsoft. Donot get scroogled by gmail. *Microsoft, https://news.microsoft.com*, 2013.

[10] Tony Mecia. Credit card issuers watch online how you shop, customize offers. *Creditcards, http://www.creditcards.com*, 2011.

[11] Jakub Mikians, L. Gyarmati, Vijay Erramilli, and Nikolaos Laoutaris. Detecting price and search discrimination on the internet. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, HotNets-XI, pages 79–84, New York, NY, USA, 2012. ACM.

[12] The Economist. Insurance data: Very personal finance. *The Economist, http://www.economist.com*, 2012.

[13] Leslie Scism and Mark Maremont. Insurers test data profiles to identify risky clients. *WSJ, http://www.wsj.com*, 2010.

[14] Katie Lobosco. Facebook friends could change your credit score. *CNN, http://money.cnn.com*, 2013.

[15] Chris Cuomo et al. Gma gets answers: Some credit card companies financially profiling customers. *ABCNews, http://abcnews.go.com*, 2009.

[16] David Mayer. Outrage as credit agency plans to mine facebook data. *Gigaom, https://gigaom.com*, 2012.

[17] H. Metwalley, S. Traverso, and M. Mellia. Unsupervised detection of web trackers. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2015.

[18] Keaton Mowery and Hovav Shacham. Pixel perfect: Fingerprinting canvas in HTML5. In Matt Fredrikson, editor, *Proceedings of W2SP 2012*. IEEE Computer Society, May 2012.

[19] Online JavaScript Obfuscator. *http://www.javascriptobfuscator.com*, 2016.

[20] JavaScript Obfuscator. *http://packer.50x.eu*, 2016.

[21] Jscrambler. *https://jscrambler.com/en/*, 2016.

[22] Javascript Obfuscate and Encoder. *http://www.jsobfuscate.com*, 2016.

[23] Alexa. *Alexa top global sites, https://www.alexa.com/topsites*, 2016.

[24] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 541–555, Washington, DC, USA, 2013. IEEE Computer Society.

[25] Pedro Fortuna. Protecting javascript source code using obfuscation. In *OWASP Europe Tour 2013 Lisbon*, 2013.