A Classification of Web Browser Fingerprinting Techniques

Randika Upathilake
Department of Systems
and Computer Engineering
Carleton University
Ottawa, ON, Canada
Email: randikaupathilake@cmail.carleton.ca

Yingkun Li
Department of Systems
and Computer Engineering
Carleton University
Ottawa, ON, Canada
Email: yingkunli@cmail.carleton.ca

Ashraf Matrawy
School of Information Technology
Carleton University
Ottawa, ON, Canada
Email: ashraf.matrawy@carleton.ca

***Abstract*—As the World Wide Web has increasingly become a necessity in daily life, the acute need to safeguard user privacy and security has become manifestly apparent. After users realized that browser cookies could allow websites to track their actions without permission or notification, many have chosen to reject cookies in order to protect their privacy. However, more recently, methods of fingerprinting a web browser have become an increasingly common practice. In this paper, we classify web browser fingerprinting into four main categories: (1) Browser Specific, (2) Canvas, (3) JavaScript Engine, and (4) Cross-browser. We then summarize the privacy and security implications, discuss commercial fingerprinting techniques, and finally present some detection and prevention methods.**

I. Introduction

When browsing a website, session state information such as the user's preferences, personal and private data may be stored on the user's device in the form of a cookie [1]. Cookies were introduced by Lou Montulli in 1994, and were designed to track a user's activities on a website [2]. While providing convenience, the ubiquitous nature of cookies has led to privacy concerns. The session state information stored in cookies can be used by websites to identify the user, track past activities or for more malicious purposes. However, the use of cookies or any active tracking methods all share one common property, which is that the identifiers can be destroyed [3]. In order to protect user privacy, modern browsers provide users with the ability to disable cookies. However, advertisers and anti-fraud companies have found a new way to track users through web browser fingerprinting [4]. The first large scale experiment on web browser fingerprinting was conducted in 2010 by P. Eckersley through the *Panopticlick* project [5]. Since then, web-based fingerprinting has captured the public's attention and once again raised privacy concerns.

Due to the presence of other forms of fingerprinting, it is important to differentiate the fingerprinting techniques classified in this paper from other methods such as *Website fingerprinting* or *Signal Fingerprinting*. In order not to confuse what we classify in this paper from other techniques, we provide the following descriptions of each of these three distinct categories of techniques:

- *Web browser fingerprinting* is the use of a set of attributes obtained from the user's browser, such as information about installed fonts or plug-ins, to uniquely identify a browsing environment without cookies or maintaining client side state [6], [7]. This is the category that we classify in this paper.
- *Website fingerprinting* on the other hand is a form of traffic analysis where an attacker observes the encrypted data and tries to draw conclusions from certain features of the traffic. This technique is used to target the protection and anonymity provided by an anonymity network such as Tor [8].
- Finally, *signal fingerprinting* is the detection of radio signal features that form a valid device fingerprint, which can then be used to make an association between observed messages and their senders [9].

This paper aims at presenting a concise overview of the web browser fingerprinting techniques in the literature to help researchers who are new to this area. Our main **contribution** in this paper is the classification of web browser fingerprinting techniques into four main categories:

1) Browser Specific Fingerprinting (Section II): is the technique where the fingerprint is dependent on the browsing environment.
2) Canvas Fingerprinting (Section III): is using the **HTML5 <canvas>** element to render an image and then read the pixel data to generate a fingerprint.
3) JavaScript Engine Fingerprinting (Section IV): is the use of JavaScript conformance tests to determine a fingerprint.
4) Cross-browser Fingerprinting (Section V): is the technique where the fingerprint is independent of the browsing environment. A subcategory of this technique is Accelerometer Fingerprinting (Section V-A). It is the use of a mobile device sensor and its calibration errors for the generation of a fingerprint.

We present the classification in an easy to understand figure (Fig.1) that summarizes the browser features used to generate a fingerprint as well as the variables which can affect the fingerprint. We include only the techniques that have been presented in the literature.

The rest of this paper is organized as follows: Sections

978-1-4799-8784-9/15/$31.00 ©2015 IEEE

II to V present the four categories. Section VI summarizes the privacy and security implications of web browser fingerprinting by discussing it's constructive and destructive uses. Section VII discusses commercially used methods and their delivery mechanisms. Section VIII presents some detection and prevention techniques applicable to different forms of web browser fingerprinting. Finally, Section IX concludes the paper.

II. Browser Specific Fingerprinting

The first large scale browser fingerprinting study was conducted by P. Eckersley in the Panopticlick project [5]. The results of the study showed that 94.2% of the users who use Flash or Java could be distinguished from one another even without the use of cookies. Browser fingerprinting uses browser-dependent features such as Flash or Java to retrieve information on installed fonts, plug-ins, the browser platform and screen resolution [1]. The results of the Panopticlick project also showed several key features of browsers fingerprinting. First is that only 15-20 bits of identifying information were necessary to uniquely identify a particular browser [5]. Second, plug-ins and fonts are the most identifying metrics, followed by User Agent, HTTP Accept, and screen resolution. The reason that plug-ins offer high entropy is due to the version numbers of the plug-ins being very precise. This stems from the fact that browser vendors prefer debuggability over defence against fingerprinting [5].

There are several weaknesses to this form of fingerprinting. The main weakness is that the fingerprint is unstable, meaning that the fingerprint can change quite easily. The instability can be caused by upgrades to the browser or a plug-in, installing a new font, or simply the addition of an external monitor which would alter the screen resolution [1]. However, it was noted in the results of the Panopticlick project that even when fingerprints changed, a simple heuristic can be used to guess when an upgrade to a previously observed fingerprint occurred [5]. Another weakness of this fingerprinting technique, and also a weakness shared by other techniques, is that it is unable to distinguish between instances of identically configured devices [3].

III. Canvas Fingerprinting

Canvas fingerprinting is a recent technique which was presented by Mowery and Shacham in 2012. It works by rendering text and WebGL scenes on to an area of the screen using the **HTML5 <canvas>** element programmatically, and then reading the pixel data back to generate a fingerprint [10]. The simple method demonstrated by Mowery and Shacham works as follows: A 2D graphics context is obtained and text or an image is drawn to the canvas. The *toDataURL(type)* method is called on the canvas object and a Base64 encoding of a PNG image containing the contents of the canvas are obtained. A hash of the Base64 encoded pixel data is created so that the entire image is not needed to be uploaded to a website. The hash is also used as the fingerprint. This technique uses WebFonts to guarantee that the correct font will always be used when drawing to the canvas. The results also showed that at least the operating system, browser version, graphics card, installed fonts, sub-pixel hinting, and anti-aliasing all play a part in the final fingerprint. Mowery and Shacham also estimate that 10-bits entropy are possible over the whole population of the web using this technique.

A 2014 study conducted by Acar et al. showed that canvas fingerprinting is the most common form of fingerprinting [11]. It was determined that approximately 5.5% of the top 100,000 websites use this technique of fingerprinting on their home pages. Surprisingly 95% of these scripts belonged to a single provider - Addthis.com. In total 20 canvas fingerprinting provider domains were found and of these, 11 were third-party scripts and 9 were in-house scripts.

This technique has the same weakness as browser fingerprinting due to the fact that the fingerprint can change across browsers. However, Canvas fingerprinting also has some very appealing properties that make it the most common form of fingerprinting [11]. As concluded in the study by Mowery and Shacham, it is consistent, it has high-entropy, it is orthogonal to other fingerprints, it is transparent to the user and it is readily obtainable [10]. Any website that is running JavaScript on a visitor's browser can fingerprint its rendering behaviour with no requirement for special access to system resources [10]. However, as with other forms of browser fingerprinting, canvas fingerprinting is unable to distinguish between users who use the exact same hardware and software [10].

IV. JavaScript Engine Fingerprinting

JavaScript Engine Fingerprinting relies on JavaScript conformance tests such as the Sputnik [12] test suite to determine a specific browser and major version number. A method demonstrated by Mulazzani et al. [13] executes a set of conformance test cases from the Sputnik test suite in a user's browser. They then compare the failed tests to a set of test cases that are known to fail in each version of a browser. This allow a match to be made to identify a specific browser and the major version. This can be considered more robust than relying on the User-Agent string to determine the browser and version number for use in fingerprinting [13]. While the User-Agent string can be modified to an arbitrary value by the user, the JavaScript engine fingerprint will be unique for each browser and cannot be ported to a different browser. Other properties of this technique are that it can be used to detect modified User-Agent strings, can be used on mobile devices, and can be used to reliably identify the browser of Tor Browser Bundle users [13].

V. Cross-browser Fingerprinting

The main differentiator between cross-browser fingerprinting and the previously discussed browser fingerprinting is that cross-browser fingerprinting uses browser-independent features to generate the fingerprint. A simple cross-browser fingerprinting algorithm can rely solely on JavaScript and use font detection as a technique [3]. Since a JavaScript engine is available and enabled by default on all modern browsers

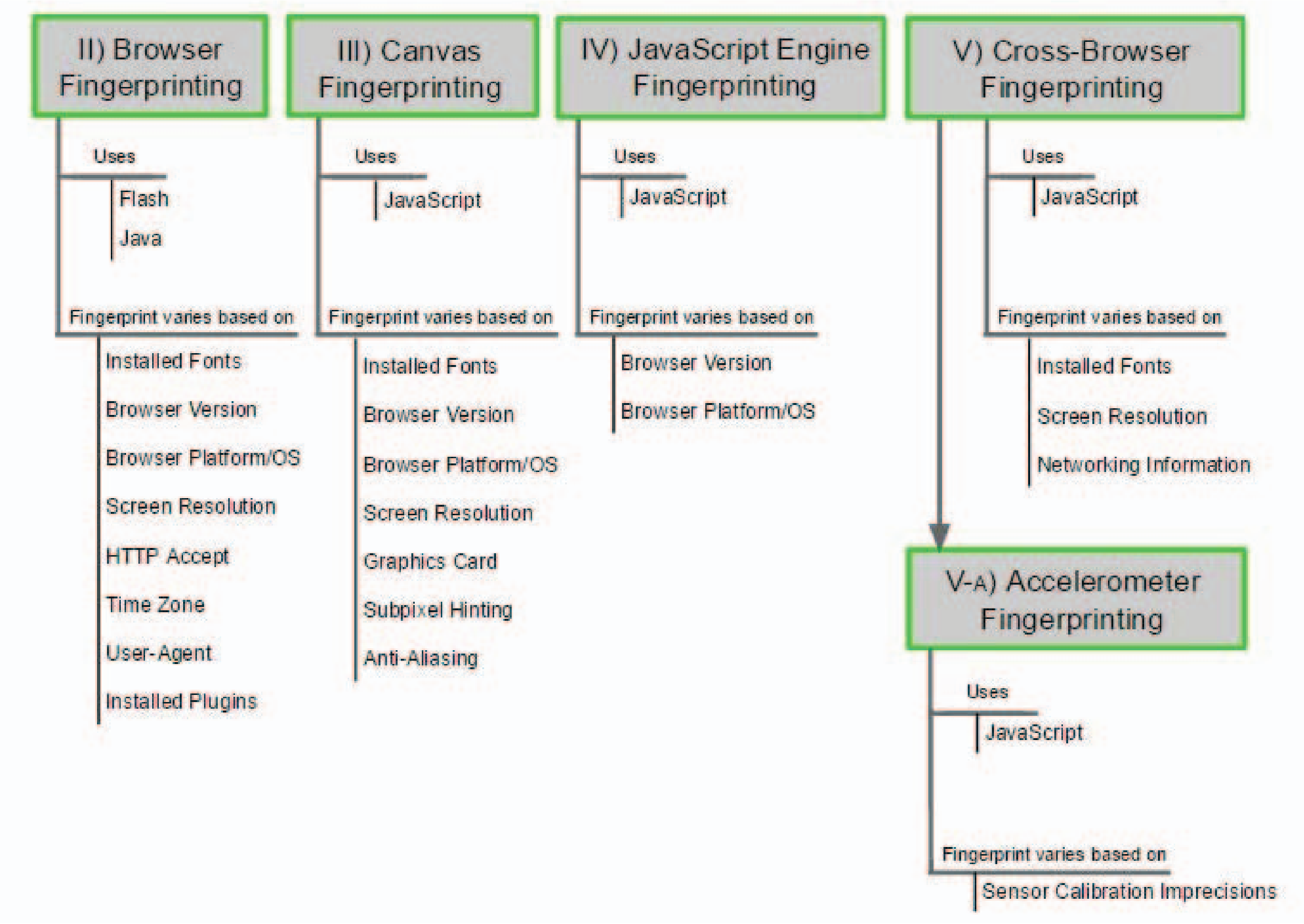


Fig. 1. Fingerprinting techniques and the attributes that affect the fingerprint.

[13], this technique differs from the Panopticlick project which relied on the browser dependent plug-ins Adobe Flash or Java for obtaining a font list [3].

Cross-browser fingerprinting shares a common weakness with the browser fingerprinting technique determined by P. Eckersley. This is the inability to distinguish between identical configurations, as is the case when multiple users browse the web in a computer lab [3].

A. Accelerometer Fingerprinting

Mobile browsers are significantly harder to fingerprint than desktop browsers. This is due to the fact that mobile browsers are more uniform and they do not have the plug-ins that are available on desktop systems [5]. A recent study by Bojinov et al. [14] showed a technique of generating a unique fingerprint by accessing a sensor that is commonly found on mobile devices. This technique allows a fingerprinting script to avoid using traditional hardware identifiers such as the IMEI and UDID [14]. The study showed that the accelerometer on mobile devices was exposed without user notification by iOS and Android browsers through JavaScript. Imprecisions in accelerometer calibration during manufacturing result in device specific measured values. The method employed by Bojinov et al. involved repeatedly querying the accelerometer and then estimating the calibration errors in each of the three dimensions to generate a unique fingerprint [14].

This form of fingerprinting can be classified as a subcategory of cross-browser fingerprinting. This is due to the fact that the generated fingerprint does not vary across browsers. Another property of this technique is that it may be exercised without user cooperation [14]. However, a factor in limiting its usefulness is that the device must be left facing up and down to gather readings to generate the fingerprint [14].

VI. Privacy and Security

Browser fingerprinting has both constructive and destructive uses [7]. However, since the technology is the same in both cases, the difference is artificial and is up to the interpretation of the user [1]. For example, the use of fingerprinting by a website to ensure that the users trying to access their services are indeed who they claim to be, instead of an attacker, can be considered a constructive use of fingerprinting. On the other hand, the use of fingerprinting by advertisers to track a user and display customized ads based on browsing habits, or by paywalls to tell users apart for paid services can be considered a destructive use [6].

There are also uses of fingerprinting that can only be considered malicious and destructive. Browser fingerprinting can be used to deliver malware that is customized for a specific browser configuration, or even a particular user in a targeted attack [15]. It can also be used to gather information on a target host [15]. This fingerprint can then be used to match against known execution environments in order to launch exploits specific to the host as well as to prevent execution inside an analysis system [15], [10]. Kolbitsch et al. examined a malicious fingerprinting script that uses a fingerprint to deliver an exploit that is likely to be successful in attacking a target's browser. It records the presence of several plug-ins such as Adobe Acrobat, Quicktime, Java, and Windows Media player. It then constructs a fingerprint and concatenates it with the browser language and then finally issues a request to a malware hosting site to fetch the malware that corresponds to the fingerprint [15].

VII. Commercial fingerprinting

Nikiforakis et al. [1] determined that there are two methods for commercial fingerprinting service delivery. The first involves first-party websites who are not involved in the fingerprinting. Instead, the fingerprinting code was delivered via advertising syndicators. In this method, the fingerprint is sent back to the fingerprinting service provider and the first-party site may not even be aware its users are being fingerprinted [1]. In the second method, the first party site contains the fingerprinting script. It fingerprints a visitor and then the fingerprint is submitted to the website in a hidden input element when the user submits credentials. The fingerprint is encrypted so it can only be decrypted by the commercial fingerprinting service provider; therefore it must be submitted to the service provider to match and return information based on the match. This allows commercial fingerprinting companies to hide their implementation details from clients [1].

The study by Nikiforakis et al. [1] also showed some interesting facts about the categories of websites that employ fingerprinting. Out of the top 10,000 websites web-crawled in their study, 15% were pornography sites and 12.5% were Personals/Dating websites. Their explanation for the findings is that the pornography sites are using fingerprinting as a way to detect shared or stolen credentials and the dating sites make use of a fingerprint to ensure that multiple profiles are not created for social-engineering purposes [1].

VIII. Detection and Prevention

The stateless nature of browser fingerprinting makes it hard to detect and even harder to opt-out. However, there are several common methods of preventing browser fingerprinting discussed in the literature. These include:

- having all browser vendors agree on a single set of API calls to expose to the web applications as well as internal implementation specifics [1];
- having blocking tools which are maintained by doing regular web-crawls to detect tracking and incorporate blocking mechanisms into the tools [11];
- the introduction of a universal font list that a browser is limited to choose from for rendering [3];
- reporting unified and uncommunicative attributes [3];
- blocking or disabling JavaScript [5];
- reducing the verbosity of the User-Agent string and the plug-in versions [5], [3];
- having Flash provide less system information and report only a standard set of fonts [1].

The following subsections summarize techniques specific to certain categories and sub-categories of the fingerprinting techniques.

A. Canvas Fingerprinting

Canvas fingerprinting is a difficult technique to automatically detect and prevent without false positives [11]. As suggested by Acar et al. [11], a solution would be to turn to crowd-sourcing. A browser tool can default to blocking all pixel data extraction attempts, but slowly over time, take into account user feedback and improve the tool to identify valid fingerprinting attempts [11].

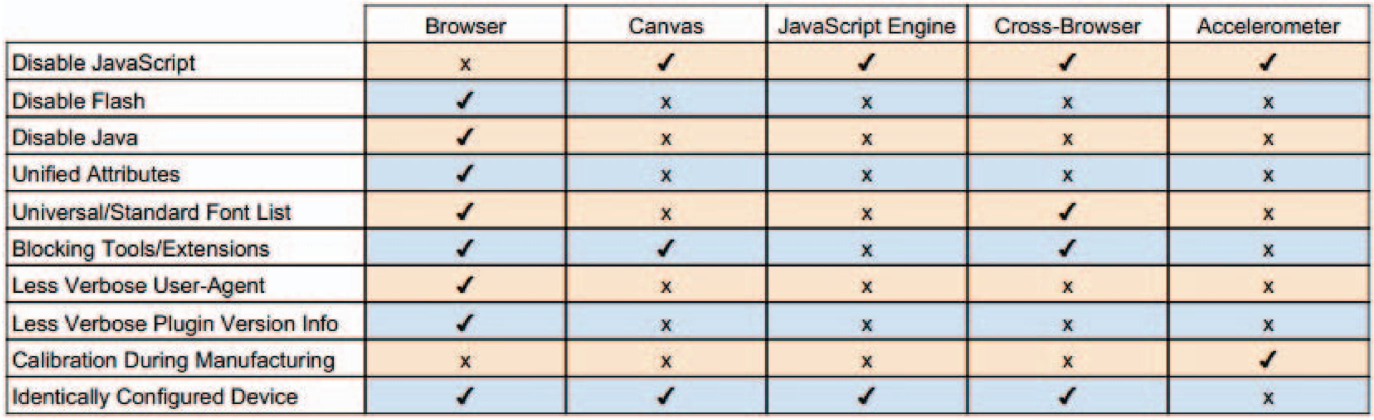
Other suggested methods for preventing canvas fingerprinting include having the browser add random pixel noise whenever pixels are extracted or having the browsers render scenes in a generic software renderer [10]. These suggestions come with a significant performance penalty and therefore, unacceptable for general use [10]. The easiest method appears to require user approval if a script requests pixel data [10]. Modern browsers already use this approach, for example with the HTML5 geolocation API [10]. However, this introduces another permission dialogue which requires user interaction.

B. Accelerometer Fingerprinting

Bojinov et al. [14] proposed two methods for the prevention of Accelerometer fingerprinting. The first proposal states that the feasibility of fingerprinting can be practically eliminated at the hardware level if the sensor in similar devices are calibrated to the same specification during the manufacturing process [14]. This would remove any variance in the sensor reading, and thus, eliminate its use for fingerprinting. The second proposal is a software based solution. A random value can be added to the sensor output at the OS level. This value can remain constant during continuous use and then changed during periods of inactivity [14].

C. Extensions and Plug-ins

The use of some common extensions and plug-ins were also mentioned in the literature. The notable ones which seemed to aid in reducing fingerprintability were AdBlock Plus [16], Ghostery [17], and NoScript [18]. Nikiforakis et al. [6] examined several highly rated extensions for Firefox and Chrome, which allowed changing the User-Agent to appear as if sessions were from different computers. They noted in their study that these were inadequate and contained discrepancies



	Browser	Canvas	JavaScript Engine	Cross-Browser	Accelerometer
Disable JavaScript	x	✓	✓	✓	✓
Disable Flash	✓	x	x	x	x
Disable Java	✓	x	x	x	x
Unified Attributes	✓	x	x	x	x
Universal/Standard Font List	✓	x	x	✓	x
Blocking Tools/Extensions	✓	✓	x	✓	x
Less Verbose User-Agent	✓	x	x	x	x
Less Verbose Plugin Version Info	✓	x	x	x	x
Calibration During Manufacturing	x	x	x	x	✓
Identically Configured Device	✓	✓	✓	✓	x

Fig. 2. Comparison of web browser fingerprinting prevention techniques

which led to the browser being more distinguishable. These discrepancies allow fingerprinting scripts to increase the accuracy of the fingerprint since they are able to tell the presence of a specific extension [1], [5]. It should be noted that these results seem to be different from what was concluded by Yen et al. in their study [19]. They suggested that changing the User-Agent string to one that corresponds to a popular browser version was a simple method to become less distinguishable [19]. The detection and prevention methods are summarized in Fig. 2.

IX. Conclusion

Web browser fingerprinting techniques have evolved rapidly since they were brought to the attention of privacy conscious users through the Panopticlick project in 2010. In this paper, we classified web browser fingerprinting techniques into four main categories. We also discussed some preventative measures that are available to reduce the likelihood of a user's web browser being fingerprinted (see Section VIII and Fig. 2).

While all the techniques have similarities and differences in their capabilities and uses, they generally share a common property (at least as concluded from the literature): they do not have the ability to distinguish between multiple users on a single device. Also, with the exception of the accelerometer based fingerprinting technique, the other techniques are not known to be able to distinguish among identically configured devices.

Acknowledgement

The authors would like to thank the anonymous referees for their valuable comments. The third author acknowledges funding from Canada's NSERC through the Discovery Grant Program.

References

[1] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting," in *IEEE Symposium on Security and Privacy*. IEEE, 2013.

[2] W. Peng and J. Cisna, "Http cookies - a promising technology," *Online Information Review*, pp. 150–153, 2000.

[3] K. Boda, A. M. Foldes, G. G. Gulyas, and S. Imre, "User tracking on the web via cross-browser fingerprinting," Information Security Technology for Applications. Springer Berlin Heidelberg, pp. 31–46, 2012.

[4] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gurses, F. Piessens, and B. Prenee, "FPDetective: Dusting the Web for Fingerprinters," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & Communications Security*. ACM, 2013.

[5] P. Eckersley, "How Unique Is Your Browser?" in *10th Privacy Enhancing Technologies Symposium*, 2010.

[6] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "On the Workings and Current Practices of Web-Based Device Fingerprinting," Security & Privacy, IEEE, May-June 2014.

[7] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham, "Fingerprinting Information in JavaScript Implementations," in *Proceedings of Web 2.0 Security & Privacy*. IEEE, May 2011.

[8] A. Panchenko, L. Niessen, A. Zinnen, and T. Enge, "Website Fingerprinting in Onion Routing Based Anonymization Networks," in *Proceedings of the 10th annual ACM workshop on Privacy in the Electronic Society*. ACM, 2011.

[9] B. Rasmussen, Kasper, and S. Capkun, "Implications of Radio Fingerprinting on the Security of Sensor Networks," in *Preeceedings of the third international conference on security and privacy in communication networks. SecureComm 2007*. IEEE, 2007.

[10] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in HTML5," in *Proceedings of Web 2.0 Security & Privacy*. IEEE, 2012.

[11] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The web never forgets: Persistent tracking mechanisms in the wild," in *Proceedings of the 21st ACM Conference on Computer and Communications Security*. ACM, 2014.

[12] Sputnik. [Online]. Available: https://code.google.com/p/sputniktests/, [Accessed May 2015].

[13] M. Mulazzani, P. Reschl, M. Huber, M. Leithner, S. Schrittwieser, and E. Weippl, "Fast and Reliable Browser Identification with JavaScript Engine Fingerprinting," in *Proceedings of Web 2.0 Security & Privacy*. IEEE, 2013.

[14] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh. (2014, August) Mobile Device Identification via Sensor Fingerprinting. [Online]. Available: http://arxiv.org/abs/1408.1416/. [Accessed May 2015].

[15] C. Kolbitsch, B. Livshits, B. Zorn, and C. Seifert, "Rozzle: De-cloaking internet malware," in *IEEE Symposium on Security and Privacy*. IEEE, 2012.

[16] Adblock plus. [Online]. Available: https://adblockplus.org/, [Accessed May 2015].

[17] Ghostery. [Online]. Available: https://www.ghostery.com/en/, [Accessed May 2015].

[18] Noscript. [Online]. Available: https://noscript.net/, [Accessed May 2015].

[19] T.-F. Yen, Y. Xie, F. Yu, R. P. Yu, and M. Abadi, "Host Fingerprinting and Tracking on the Web: Privacy and Security Implications," in *Proceedings of the 19th Annual Network and Distributed System Security Symposium*. Internet Society, 2012.