Web-based Fingerprinting Techniques

Vítor Bernardo and Dulce Domingos
LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Lisboa, Portugal

Keywords: Browser Fingerprinting, Cross-browser Fingerprinting, Device Fingerprinting, Privacy, Fingerprint.

Abstract: The concept of device fingerprinting is based in the assumption that each electronic device holds a unique set of physical and/or logical features that others can capture and use to differentiate it from the whole. Web-based fingerprinting, a particular case of device fingerprinting, allows website owners to differentiate devices based on the set of information that browsers transmit. Depending on the techniques being used, a website can track a device based on its browser features (browser fingerprinting) or based on system settings (cross-browser fingerprinting). The latter allows identification of the device even when more than one browser is used.
Several different works have introduced new techniques over the last years proving that fingerprinting can be done in multiple ways, but there is not a consolidated work gathering all of them. The current work identifies known web-based fingerprinting techniques, categorizing them as which ones are browser and which are cross-browser and showing real examples of the data that can be captured with each technique. The study is synthesized in a taxonomy, which provides a clear separation between techniques, making it easier to identify the threats to security and privacy inherent to each one.

1 INTRODUCTION

Device fingerprinting is based on the assumption that no two devices are exactly alike and that profiles can be created by capturing the emanation patterns sent or leaked from the devices, as long as these externalizations[1] are repetitive through time. Small physical differences in the components of the devices, that were introduced during the manufacturing process, may result in slightly different behaviours and externalizations, which could be captured and used to create a fingerprint (Jenkins et al., 2014). Other examples of fingerprinting could be the tracking of clock deviation between the internal clock of a client's device and the clock of a server (Kohno et al., 2005) or the collection of the set of fonts and browser plugins installed on a system (Eckersley, 2010).

Web-based fingerprinting, a particular case of device fingerprinting, allows a website owner to track devices' accesses throughout time, in an almost invisible way. In fact, even if the user is aware of privacy issues and takes precautions, whether by actively deleting cookies, blocking all cookies or using a browser in "private mode", his device will still be fingerprintable. This makes the use of web-based fingerprinting far more upsetting than simple cookies.

In most cases, web-based fingerprinting is used to track users activity in sites and bind a device fingerprint to a user profile (together with its preferences, tastes and interests). The interest of advertising companies in this kind of information is foreseeable, as it allows them to adjust the publicity to the users interests.

Indeed, in 2014, the Article 29 Data Protection Working Party, a European Union advisor on data protection, stated that: " (...) fingerprint provides the ability to distinguish one device from another and can be used as a covert alternative for cookies to track internet behaviour over time. As a result, an individual may be associated, and therefore identified, or made identifiable, by that device fingerprint. (...) The data protection risks of device fingerprinting are increased by the fact that the unique set of information elements is not only available to the website publisher, but also to many other third parties." (Article 29 Data Protection Working Party, 2014) (p. 6).

Web-based fingerprinting relies in capability to collect information about the device's operating system (OS) properties, installed software, and other logical configurations to get a unique signature from the device - rather than trying to infer patterns from the behaviour of the equipment, such as the "hardware-

[1]The expression "externalization" refers to any display of activity emanating from a certain device.

Bernardo, V. and Domingos, D.
Web-based Fingerprinting Techniques.
DOI: 10.5220/0005965602710282
In *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT*, pages 271-282
ISBN: 978-989-758-196-0

based" device fingerprinting approach would. Web-based fingerprinting does not require special equipment or a specific scenario to be put into practise.

Two different approaches can be taken regarding the type of client-side features that will be processed to extract a signature: browser or cross-browser fingerprinting. Browser fingerprinting relies on the distinctive features of the client browser and/or OS and its purpose is to create a unique signature based on the information collected from the pair (browser, OS). Cross-browser fingerprinting, on the other hand, is based only in non-browser features, what makes it resilient to the use of multiple browsers by the same device. This approach requires system settings to be collected such as, OS version, CPU information, network interfaces information, number of processors, screen size, etc.

The next section describes how web-based fingerprinting works. Section 3 depicts the most prominent works addressing web-based fingerprinting techniques. The bulk of the document is focused in the analysis of the techniques (section 4), together with the results of the tests using different browsers. The taxonomy of web-based fingerprinting techniques can be found at the end of this section. Finally, section 5 concludes the paper, discussing the threats to security and privacy inherent to web-based fingerprinting and possible mitigations.

2 HOW WEB-BASED FINGERPRINTING WORKS

In a typical web-based fingerprinting scenario, the client's browser requests the webpage code to the server (through an HTTP request), in order to render it for the user. Within this request, some fingerprinting information from the user's system can already be collected, namely the information that is sent in the User-Agent - the HTTP header field that the most popular browsers use to indicate the browser and OS versions. In the HTTP response, the web server can include de client-side fingerprinting script. Most fingerprinting techniques are based in client-side code execution because this type of technology allows the browser to make direct calls to the OS or other machine configurations and to send that data back to the webserver asynchronously (without interfering with the display and behaviour of the existing page).

After the browser has processed the script and sent the result to the web server, the later computes a unique identifier based on the information received and stores it in a database - this will be the fingerprint for this user's device. Although this identifier could be seen as a simple hash of the device's properties, this would soon prove to be a simplistic approach as it would render the fingerprint useless at the slightest change in the client's environment. A better approach would be to store an array of the device's properties. From this moment on, the website owner can track that device throughout the pages it visits, without the use of cookies, as long as they all contain the fingerprinting script.

It is important to note that the tracking can be done among different sites or domains, as long as all the websites share the same fingerprinting database.

In a different model of fingerprinting, the agent doing the tracking might not be the website provider but a third-party agent. This scenario is described in (Nikiforakis et al., 2013). A site owner can agree with an advertisement company the deployment of a content from the later in the webpage, which visitors will retrieve unknowingly, containing a fingerprinting script. This version presents a more disturbing scenario: the fingerprinting agent can track users on multiple third-party websites, as long as the site owners agree to include the advertisement. This allows a more intrusive profiling of the web users as different kinds of websites (with different subjects) provide a richer view of the individual.

3 RELATED WORK

As far as we are aware, (Mayer, 2009) was the first author to present a fingerprint based on browser plugins and mime-types. (Eckersley, 2010) also discusses these techniques and introduces other cross-browser approaches, such as font detection and OS features. This work paved the way for web-based fingerprinting and was the basis for the creation of Electronic Frontier Foundation's tracking awareness website, *Panopticlick*[2].

Fingerprinting based on HTTP Header Fields is one of the most explored techniques and (Eckersley, 2010), as well as (Mowery et al., 2011), (Nikiforakis et al., 2013), and (Acar et al., 2013) discuss this technique.

The study that (Nikiforakis et al., 2013) present is the one that shares most resemblance with the present work. It describes some of the practices of device identification through web-based fingerprinting techniques available by then and measures the adoption of fingerprinting on the web. The authors also present a taxonomy for the fingerprinting techniques found in "three large, commercial companies" along with *Pan-*

[2]Available at https://panopticlick.eff.org

opticlick. Although very enlightening, the study is limited to the techniques that the three software manufacturers and the *Panopticlick* website used at the time. Therefore, other techniques, such as HTML5 canvas fingerprinting or exploitation of DNS leaks, are not covered.

(Khademi et al., 2015) show the effectiveness of some of the techniques depicted in the current work, by implementing a web-based hybrid fingerprinting tool: Fybrid.

Focusing on mitigation techniques, (Nikiforakis et al., 2015) propose a tailored browser (*PriVaricator*) which makes every visit appear different to a fingerprinting site, resulting in a different fingerprint for each visit. We also discuss mitigation techniques in section 5.

4 TECHNIQUES FOR WEB--BASED FINGERPRINTING

This section describes our study on web-based fingerprinting techniques. The analysed techniques were gathered from multiple sources, ranging from published papers to websites used with a fingerprinting purpose - mostly, websites that show how web-based fingerprinting is possible. These sources are referenced along this section.

In our tests, we used different browsers: *Mozilla Firefox 38.0.1*; *Microsoft Internet Explorer 10.0.9200.17357* (MS IE); *Google Chrome ver. 43.0.2357.81 m*; *Opera/9.80 (Windows NT 6.2; WOW64) Presto/2.12.388 Version/12.17*; *QtWeb Internet Browser 3.8.5 (build 108)*; *Midori 0.5.10* and *Chromium Portable version 44.0.2383.0*.

After analysing the techniques, our results in a taxonomy are synthesized in a taxonomy.

4.1 IP Address

The use of network mechanisms that changes the user's external IP address (e.g. NAT systems or proxies) makes this information unreliable for tracking purposes. Still, while not providing enough information to create a fingerprint, the IP address can help adjusting and completing other methods of fingerprinting. IP addresses are sent in the Internet Layer of the TCP/IP model so, in theory, the website always receives this information. The collection of the IP addresses provides information to perform cross-browser fingerprinting.

Advantages: The IP address of the originator is sent for every HTTP request without requiring any special scripting from the website owner.
Disadvantages: With the widespread use of NAT, the tracking of a device through its IP address became unfeasible. Moreover, the ever-growing use of TOR networks and IP spoofing methods demands caution when associating an IP address to a device.

This technique has two variations that are described next.

a) Local IP Address (IPv4): The website *IPLeak.net*[3] allows checking if the client browser uses the WebRTC API. WebRTC is a browser-to-browser application for voice calling, video chat, and P2P file sharing. WebRTC implements STUN (Session Traversal Utilities for Nat)[4], a network protocol that allows an end host to discover the public IP address even when it is located behind a NAT. With information about the client's local IP address, it is possible to verify if two different signatures with the same external IP address correspond to two different devices behind a NAT.

Advantages: Allows to distinguish different devices behind a NAT.
Disadvantages: Some browsers lack the native support of WebRTC, what makes them resilient to this technique. At the time of writing, Microsoft's Internet Explorer was one of them.

b) DNS Leaks: When a DNS request is made to the default DNS servers (usually belonging to the internet service provider), instead of the anonymous DNS servers assigned by the anonymity network, it is considered a "DNS leak". In this technique, the website generates a certain number of non-existent second level domain names and includes them in the code of the requested page. When the client browser finds the references to these domains in the page code it tries to resolve them, by querying its DNS servers. The DNS servers, in turn, will not have any cached registry for those domains (because they do not exist), so they will request for addresses to the domain authority. Once the DNS servers detect the call to the fake domains they forward the requests to the *fingerprinter*, together with the names of the DNS servers requesting them. Then, the *fingerprinter* can correlate the faux domains with the IP address that requested the page, and associate an array of DNS servers to that profile.

Advantages: The data collection does not rely on any scripting, making it more effective.

[3]Available at https://ipleak.net
[4]See http://www.voip-info.org/wiki/view/STUN

Disadvantages: Requires access to the domain authority DNS servers.

4.2 HTTP Request Header Fields

The HTTP protocol specification describes a series of request header fields (Fielding and Reschke, 2014). Because HTTP header fields are a well-known source of information, many authors have mentioned them in their works ((Eckersley, 2010), (Nikiforakis et al., 2013), (Acar et al., 2013), (Mowery et al., 2011)).

Next, we describe four of the HTTP Header fields that are used to gather information about the client's system. Advantages and disadvantages are presented in the end of the subsection, as all of them share the same features.

a) Accept Field: This field allows the browser to inform the server about the Content-Types that are acceptable for the response - the media types that the browser understands and how well it understands them. This field provides browser related information and can contribute to the browser fingerprinting. Table 1 shows the values sent by 3 different browsers.

Table 1: Three different Accept field values.

Browser	Accept value
Mozilla Firefox	text/html, application/xhtml+xml, application/xml;q=0.9,*/*;q=0.8
Google Chrome	text/html, application/xhtml+xml, application/xml;q=0.9, image/webp, */*;q=0.8
MS IE	text/html, application/xhtml+xml, */*

b) Accept-Encoding Field: Compliant browsers should announce to the server what methods they support before downloading the correct format. This field provides browser related information and can contribute to the browser fingerprinting. Table 2 shows 3 possible values for this field.

Table 2: Three different Accept-Encoding field values.

Browser	Accept-Encoding value
Mozilla Firefox	gzip, deflate
Google Chrome	gzip, deflate, sdch
QtWeb	gzip

c) Accept-Language Field: A webpage might have the same content available in several languages. The language can be selected automatically by the server, based on the preference indicated in this field. This property also contributes to browser fingerprinting. Table 3 shows 3 different values for the Accept-Language field.

Table 3: Three different Accept-Language field values.

Browser	Accept-Language value
QtWeb	pt-PT,en,*
Google Chrome	en-US,en;q=0.8
MS IE	pt-PT,pt;q=0.8,en-GB;q=0.5,en;q=0.

d) User-Agent Field: This field can provide specific information about the browser brand and version. It also allows inferring the type of OS, which makes it useful for browser fingerprinting and cross-browser fingerprinting at the same time. Table 4 shows 3 different possible values for this field.

Table 4: Three different User-Agent field values.

Browser	User-Agent value
Mozilla Firefox	Mozilla/5.0 (Windows NT 6.2; WOW64; rv:38.0) Gecko/20100101 Firefox/38.0
MS IE	Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0)
Google Chrome	Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.81 Safari/537.36

Advantages: The biggest advantage of using HTTP headers is that every incoming request contains a set of them, making it easier for the webserver to collect the information. Therefore, no client-scripting is needed.
Disadvantages: The HTTP fields do not provide enough diversity to create a unique signature. This technique must always be used complementary to others. Moreover, a user with a customized browser could clean or modify the header files, affecting the fingerprinting.

4.3 Browser Properties

System settings such as the type of contents supported by the browser, the local clock time or the client's system default language can be externalized by the browser software. We discuss advantages and disadvantages in the end of this subsection, as they are common for both techniques.

a) Plugin and Mime-type Enumeration: When a website needs to query if a certain plugin exists in the clients system, to properly display/run some type of content, it can retrieve an array of plugins by accessing to JavaScript's `navigator.plugins`. This technique is described in the following works: (Eckersley, 2010), (Nikiforakis et al., 2015), (Acar et al., 2013) and (Mayer, 2009).

This information can be used for both browser fingerprinting and cross-browser fingerprinting.

An example of the information we can get with this technique is: *Plugin 0: Adobe Acrobat; Plugin 2: Google Update; Plugin 3: Java Deployment Toolkit 7.0.450.18*.

b) Do-Not-Track: The Do-Not-Track header was a proposed HTTP header field (DNT) with the objective of increasing the privacy of the webpage users. The header contains a flag that indicates whether the client is willing to be tracked across websites. Unfortunately, the browser user has no control over whether the request is honoured or not, so the effectiveness of this measure is arguable.

The Do-Not-Track header can be retrieved using the JavaScript language and calling the value `navigator.doNotTrack`.

(Mayer and Mitchell, 2012), (Acar et al., 2013) and (Nikiforakis et al., 2013) mentioned the Do-Not-Track header and agreed that its usefulness is questionable, at best. Because it produces slightly different answers across browsers (see Table 5 below), this technique can add diversity to the browser fingerprinting.

Table 5: Three different Do-Not-Track field values.

Browser	Do-Not-Track
Mozilla Firefox	unspecified
MS IE	undefined
Google Chrome	null

Advantages: The retrieval of the browser's properties provides an extensive list of browser related information, including plugin versions, which adds diversity to the profiling process.
Disadvantages: The browser plugins are pieces of software that can be removed or updated (to new versions). The creation of a signature should not rely on a simple hash of the plugin list, or else the fingerprint would be rendered useless right after the first plugin update. Also, browsers react differently to the `navigator.plugins` request.

4.4 Browser Behaviour

Browser vendors are free to include their own logic in the code that is embodied in their software. This means that, the outcome of a certain operation when performed by two different browsers might be the same, although their underlying mechanisms are different. In this section we describe different browser behaviour techniques.

a) Browser Rounding and Fractional Pixels: The way that different browser handle math calculations, and rounding in particular, can be used to identify the client's browser and contribute to the fingerprinting process.

The webpage "Browser Rounding and Fractional Pixels"[5] contains a test page where users can test their browser's behaviour. The site calculates percentages of decimal values for multiple graphic boxes with fractional pixels. Being a browser benchmarking technique, this approach can contribute to perform browser fingerprinting. The different results for browser rounding and fractional pixels of an element (*Box2*) are shown in Table 6.

Table 6: Three different results for browser rounding and fractional pixels.

Browser	Calculated width for *Box2*
Mozilla Firefox	666.8499755859375px
MS IE	667px
Google Chrome	666.859375px

Advantages: Unlike other browser behaviour-related approaches that are affected by the system usage at a given time (e.g. CPU load or allocated RAM memory), this technique is unaffected by such constraints.
Disadvantages: This approach is not enough to create a unique signature. At most, it can help identifying the browser, but it requires a comparison database to relate the measured patterns with the browsers' brands and models.

b) Canvas Fingerprinting: The canvas fingerprinting technique applies a principle similar to the previously shown: the rendering of a graphical object by different systems (browsers) produces different output, and therefore different fingerprints. (Mowery and Shacham, 2012) show that the rendering of fonts and graphical elements have slight variations between different browsers, what allows the extraction of an individual signature.

In their study about canvas fingerprinting and evercookie feasibility, (Acar et al., 2014) crawled the Top Alexa 100,000 sites and found that "more than 5.5% of crawled sites actively ran canvas fingerprinting scripts on their home pages."

The website *BrowserLeaks.com*[6] provides a page to test canvas fingerprinting, but warns that the comparison database, used for matching the incoming signatures, is not complete enough to provide a complete coverage of the whole universe of browsers and does not collect new signatures.

[5]Available at http://cruft.io/posts/percentage-calculations-in-ie
[6]Available at https://www.browserleaks.com/

This technique can contribute to perform browser fingerprinting. The results of a test with 3 different browsers can be seen in Table 7.

Table 7: Three different browser signatures built from canvas fingerprinting.

Browser	Signature	Website general conclusion
Mozilla Firefox	5525E5D4	"It is very likely that you are using [Firefox] on [Windows]"
MS IE	62939B59	"It is very likely that you are using [Internet Explorer] on [Windows]"
Google Chrome	F921F32B	"Your system fingerprint appears to be unique (...)"

Advantages: Similarly to "Browser rounding and fractional pixels", this technique is not affected by the computing workload at the moment of the data collection.
Disadvantages: The technique is only reliable when backed up by a database that maps the whole browser signature universe and that is constantly collecting new signatures in order to be updated.

c) Browser Performances: (Mowery et al., 2011) presents a technique for measuring timing differences of multiple operations in the core of the JavaScript language, which can contribute to perform browser fingerprinting. According to the authors, it would be possible to distinguish not only browser versions but also micro architectural features not normally exposed to JavaScript.

The results retrieved from the website *V8 Benchmark Suite*[7] can be seen in Table 8.

Table 8: Example of JavaScript Performance Benchmarking for the Mozilla Firefox browser, retrieved from the website V8 Benchmark Suite.

Benchmarking script	Test#1 score	Test#2 score	Test#3 score	Test#4 score
Richards	5790	6023	6051	5871
DeltaBlue	10731	20556	13052	1950
Crypto	4957	4556	7585	4404

Advantages: Does not rely on information that is communicated by the browser. It rather observes the behaviour and registers the results. This makes the job harder for those trying to develop anti-fingerprinting browsers, because the simple mitigation of the data sent is not enough to protect against this technique.
Disadvantages: The performance information collected from the user's system is highly dependable on the processing being done at that moment and, therefore, different tests might show large discrepancies in the time values. On other hand, the authors of this work agree that " [o]ne of the largest weaknesses in [the] approach is that the fingerprinting time is very large - usually over 3 minutes " (p. 3). In fact, while the tests are being performed, the whole client system is affected by a sudden degradation of performance, mostly noticed by the browser's lack of responsivity. This undermines any possibility of using this technique on a stealthy way.

4.5 Operating System Features

System related settings, such as the OS version, the amount of available RAM or the current system time are features that can also be retrieved to create cross-browser fingerprints.

Java and Flash plugins are known to access system settings in a fashion that is not totally privacy-friendly, as they bypass the browser's controls. The TorProject team (developers of the Tor Browser), for instance, warns their users about the threats to anonymity associated with the use of Java and Flash plugin.[8]. The collection of OS related information using these technologies is mentioned by (Eckersley, 2010) and (Nikiforakis et al., 2013), although not extensively debated. The universe of information possible to retrieve from the system with the Java technology is significantly greater than the one provided by Flash.

a) Using Flash Plugin: The online ActionScript 3.0 Reference for Adobe Flash Platform website[9] informs that "[t]he Capabilities class provides properties that describe the system and runtime that are hosting the application. (...) By using the Capabilities class to determine what capabilities the client has, you can provide appropriate content to as many users as possible".

The data collection through the Flash plugin allows browser fingerprinting and cross-browser fingerprinting, because there is plugin-related information (browser configuration) and system information being retrieved. Table 9 shows a subset of the results retrieved from the site *BrowserLeaks.com*[10].
b) Using Java: The Java plugin can provide information about the local Java Virtual Machine (JVM). This technique provides system-related information, therefore allowing to perform cross-browser fingerprint-

[7]Available at http://v8.googlecode.com/svn/data/ benchmarks/v7/run.html

[8]Available at https://www.torproject.org/docs/faq.html.en

[9]Available at http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/index.html

[10]Available at http://https://www.browserleaks.com

Table 9: Information retrieved from the website *BrowserLeaks.com* for the Mozilla Firefox and Google Chrome.

Plugin info	Mozilla Firefox	Google Chrome
Flash Version	Shockwave Flash 17.0 r0	Shockwave Flash 18.0 r0
Plugin filename	NPSWF32 _17_0_0_188.dll	pepflashplayer.dll

ing. Table 10 shows a subset of the Java plugin data retrieved from the website *BrowserLeaks.com*, for the browsers Internet Explorer and Midori.

Table 10: Information retrieved from the website *BrowserLeaks.com* for the Internet Explorer and Midori browsers.

JVM info	MS IE	Midori
JVM Uptime	1999278	1871881
JVM Start Time	1435274207429	1435274336050
Compilation Time	334	285

Advantages: This technique provides a fair amount of system-related information that is collected by the plugins, via API calls to the OS. Java plugin returns the information it reads from the JVM which, by default, is common to all Java-dependable software on a particular system.
Disadvantages: The approach using Java is the one that provides the most verbose information (i.e. actual system data, instead of Flash's Boolean properties), making it more suitable for creating a system signature. However, because nowadays most of the browsers are currently prompting users about whether they are sure to run Java Applets, it is difficult to collect this information in a stealthy fashion.

c) Clock Skew: (Kohno et al., 2005) suggested the fingerprinting of devices through comparison of the clock skew between the client system and the server clock. According to the authors, it would be possible to exploit deviations in clock skews "even when the measurer is thousands of miles, multiple hops, and tens of milliseconds away from the fingerprinted device, and when the fingerprinted device is connected to the Internet from different locations" (p. 1). The authors stressed that "one can use [the] TCP timestamps-based method even when the fingerprintee's system time is maintained via NTP". This technique allows to perform cross-browser fingerprinting.

Advantages: This technique allows the identification of different devices behind a NAT and even the detection of virtual honeynets (the performed test showed that virtual machines did not have constant, or near constant, clock skews).
Disadvantages: The measuring duration of the tests showed in the work of (Kohno et al., 2005) range up to 120 minutes which is a rather high expectancy time for a user to be browsing on a certain domain. Moreover, the authors show the limitations of this technique when they state that the technique does "not provide unique serial numbers for devices, but (...) skew estimates".

4.6 Hardware Features

Browsers can access hardware settings, such as the screen resolution, number of CPU cores, the amount of dedicated memory or the identification of network interfaces. Although the combination of the above settings is far from being able to provide a unique set, once combined with browser settings and other fingerprinting methods they can contribute to the creation of a unique signature.

a) Screen Properties: The retrieval of the screen properties is possible to achieve through JavaScript or Flash. The use of JavaScript's `window.screen` object allows the retrieval of basic information about the screen of the client. It is possible to gather information about screen height (in pixels), width, and bit depth of the colour palette available for displaying images. This type of data collection was addressed in (Acar et al., 2013) with the use of a fingerprinting detector, the FPDetective.

An example of screen properties can be seen in Table 11, taken from the website *BrowserLeaks.com*.

Table 11: Screen properties for three different browsers, taken from the website *BrowserLeaks.com*.

Property	Mozilla Firefox	Midori	QtWeb
availHeight	860	900	860
colorDepth	24	24	32
pixelDepth	24	24	32

Advantages: The collection of screen size values allows the *fingerprinter* to infer if the client's device is a smartphone, a tablet or a two-screen desktop computer (the widths of both screens would be added).
Disadvantages: As shown in the example, caution must be taken when processing these values. Different browser implementations can change the measuring rules, therefore producing different signatures and jeopardizing any cross-browser fingerprinting attempt.

b) CPU and RAM Memory: The Java technology allows to retrieve information about the number of available processors, and the amount of free, max and total RAM memory. Browsers showed different behaviour for this data retrieval: while Internet Explorer, Mozilla Firefox and Midori provided the requested information, others returned no data at all. This technique contributes to cross-browser fingerprinting, as Java accesses OS interfaces.

Table 12 shows the different values for CPU and RAM settings retrieved using a Java plugin with the Internet Explorer and the Mozilla Firefox browsers.

Table 12: Examples of different results for (same) CPU and RAM settings.

CPU and RAM	Internet Explorer	Mozilla Firefox
Free Memory	8.135.728	10.363.232
Max Memory	259.522.560	259.522.560
Total Memory	16.252.928	16.384.000

Advantages: This technique provides very specific system-related information that might help characterize the type of web user (e.g. a tuned up system might indicate a gamer or a graphics designer), which might be interesting information for website owners or advertisers.
Disadvantages: Properties like “Free memory” or “Total memory” should not be relied upon, as they are not consistent.

c) Network Interfaces Enumeration: Java enables the retrieval of the name and status of physical and virtual networks. Among the type of information returned is the name of the interface, the status (i.e. if the interface is enabled or not), the Maximum Transmission Unit (MTU), if it is point-to-point, if it supports multicast, if it is loopback, and if the interface is virtual.

The information retrieved with this technique is system-related, therefore it is suitable for cross-browser fingerprinting. Table 13 presents one example of the 106 network interfaces retrieved from *BrowserLeaks.com*

Table 13: One of the 106 network interfaces retrieved from the website *BrowserLeaks.com* for the Internet Explorer browser.

VMware Virtual Ethernet Adapter for VMnet8				
Interface	Is Up	MTU	Point-To-Point	Multicast
eth41	true	1500	false	true

Advantages: This technique is able to quickly identify if the device being used at the current time is the same used before. This feature can have practical application in situations where a website wants to make sure that the user that signs in using a different hardware from the usual is, in fact, the real user. In case of an incoming new signature, the website can prompt the user with a security question or request a multi-channel authentication.
Disadvantages: The hardware’s collectable information is slim and there is not enough diversity in order to build a unique signature. Therefore, fingerprinting by collection of hardware features must rely on additional techniques in order to be effective.

4.7 Font Detection/Enumeration

Different OSs have their own font sets. On top of that, software suites such as Microsoft Office or Adobe Creative Suite usually add their own set of fonts to the system. Additionally, users can add their own fonts to the system, as well. The resulting set of fonts can be diverse enough to build a signature or, at least, allow the identification of the OS. (Boda et al., 2012) show how the universe of fonts is affected by the installation of Office and Adobe suites. Fingerprinting by font enumeration is mentioned in the works of (Eckersley, 2010), (Nikiforakis et al., 2013), and (Acar et al., 2013), making this one of the most commonly mentioned fingerprinting techniques, together with plugin detection. Fonts can be enumerated using JavaScript, Flash, Silverlight or Java, with different levels of efficiency.

Because system fonts are managed at the OS level, this technique allows to perform cross-browser fingerprinting. As stated in the website *latit.lab*[11], the font detection done through JavaScript and CSS takes advantage of the fact that “each character appears differently in different fonts. So different fonts will take different width and height for the same string of characters of same font-size”. By measuring the output, the *fingerprinter* is able to identify if the rendered font is different from the fall back and, if that case, resident in the system. Table 14 shows 3 different results for different browsers.

Table 14: Font detection performed by the website *lalit.org* for three different browsers.

Browser	JavaScript and CSS	Flash
MS IE	276 of 512 font	276 fonts
Google Chrome	231 of 512 fonts	270 fonts
Opera	264 of 512 fonts	276 fonts

Advantages: The enumeration of fonts is, possibly, the data collection technique that provides the biggest

[11]Available at http://www.lalit.org/lab/javascript-css-font-detect

amount of data from the client's system. Although the list of font sets can change (e.g. fonts might be actively added by the user or added because new software was installed), the *fingerprinter* can still find this risk negligible and decide to create a signature from the list of fonts. OSs can also be inferred from the font sets, as long as the *fingerprinter* has a database of specific OS featuring fonts. The same applies for browsers.
Disadvantages: In order to perform the fingerprinting using JavaScript, a database of fonts is needed for comparison. A complete identification of the fonts depends on having the most extensive database of fonts possible, to encompass all fonts that might possibly appear.

4.8 Cached Objects

The technique of using stored objects on the client side as a way to uniquely identify a device is fairly similar across the technologies that have this ability: first the *fingerprinter* checks the cached object holder for the presence of any previous unique ID (stored from a former visit). If no ID exists the website uses the cached object to store a somewhat random but unique ID that will allow further visits to be tracked and linked to this device. Flash and HTML5 have their own mechanisms for storing data, which are described furthermore.

Strictly speaking, the exploitation of cached content cannot be considered a fingerprinting technique. After all, devices are not being identified by a set of unique features but rather, by a distinctive mark that has been previously etched in each one. However, for the unique identifier to be stored in the system in the first place, it was necessary to perform a previous verification of "non-existence" of that device in the collected universe, which is, by itself, a fingerprinting-like processing.

a) HTTP Cache: The caching of webpage contents allows to reduce bandwidth usage, waiting times and server load. When requesting a resource it has downloaded and stored previously, the client browser indicates the version of the stored object and asks the server if it should download a newer one. An exploitation for this behaviour was described in (Zalewski, 2012): if the server has the chance to indicate the client's browser that it should or not replace a file already stored on its system, then it has a certain degree of control over that file and can use it as an ID for tracking purposes.

Advantages: Most browsers allow the storing of objects by default, especially when used in mobile devices, making this technique reliable for *fingerprinters*. Systems do this to reduce bandwidth usage and to increase access speed, as stated before.
Disadvantages: Devices configured not to store cache or to delete cached objects when the browser session is over are not affected by this technique, although those that allow the storing during session could be tracked throughout that period.

b) Browsing History: The collection of the URLs in the browser's history is done by using the technique described in (Janc and Olejnik, 2010), which allows "an attacker to determine if a particular URL has been visited by a client's browser through applying CSS styles distinguishing between visited and unvisited links. (...) the attacker must supply the client with a list of URLs to check and infer which links exist in the client's history by examining the computed CSS values on the client-side" (p. 4).

Because the creation of a signature of the browsing history would be rather useless, the authors decided to create "history profiles" (that act as fingerprints), where URLs are categorized by website subject (e.g. Shopping, Entertainment, Travel, Vehicles, etc.). In (Olejnik et al., 2012), the authors state that "a large number of users have unique personal browsing interests even when analysed using the more coarse-grained category metric (...) In a real scenario of an advertising provider, multiple repetitions of each category in the profiles are likely used to enumerate the strength of interest in the category which provides additional information" (p. 9).

This test was performed in a universe of 368.284 web histories, and more than 69% of users had a unique fingerprint. This technique can contribute to browser fingerprinting.

Advantages: This technique is a very appealing approach for advertisers, as they get more information with less processing on their side (i.e. if browsing profiles are already categorized into website subjects, they do not have to associate signatures with hits).
Disadvantages: (Olejnik et al., 2012) "believe that Web browsing preferences can be used as an efficient behavioural fingerprint which is in many cases stable over time" (p. 14). The question is how long can it take to have a stable browsing fingerprint? The technique assumes that the user accesses the website frequently, creating temporary profiles but, for sporadic users this approach is not reliable. On other hand, if a device is shared by multiple users using the same account, the technique might not be able to converge the patterns into a single profile, due to the different browsing behaviour of each user.

c) Local Shared Objects (Flash Cookies): Using previous versions of Flash, developers could save information between sessions by using "normal" cookies, but the process was considered difficult for developers to implement - creating a cookie requires the use of a language outside Flash (like JavaScript or ASP). In the Flash MX version, Macromedia introduced the Local Shared Object (LSO), which provides an easier way to store information (i.e. only requires the use of ActionScript).

LSOs provide the only method by which a Flash application can store information on a user's computer. Intended uses of the object include storing a user's name, a favourite colour, or the progress in a game.

Works of (Nikiforakis et al., 2013), (Mayer and Mitchell, 2012), and (Acar et al., 2014) show how LSO can be used to track users, by performing a browser fingerprinting.

The Electronic Privacy Information Centre (EPIC) warns for the risks of identification of individuals in an article regarding Local Shared Objects[12]. According to EPIC "the Flash movie can create a unique ID and store that ID in a Flash cookie on a user's computer. The Flash movie can then communicate this information to a database, or other applications. Subsequent visits of the same users could be tracked by reading the ID contained in the Flash cookie".

Advantages: Flash cookies are a powerful way to track users because they are still not properly addressed by browsers and their management is not trivial (i.e. management is not done together with HTTP cookies). This lack of proper management paves the way for exploiting this functionality for tracking or fingerprinting.
Disadvantages: Because it requires the storing of information, this technique is considered intrusive. The use of this mechanism must abide to Article 5(3) of Directive 2002/58/EC, amended by the Directive 2009/136/EC (also known as the ePrivacy directive), which requires prior informed consent for storage or access to information stored on a user's equipment.

d) Web Storage (HTML5 Cookies): HTML5 introduced two related mechanisms, similar to HTTP session cookies, for storing name-value pairs on the client side: `sessionStorage` and `localStorage`. According to the *HTML Living Standard*[13]: "Storage object provides access to a list of key/value pairs, which are sometimes called items".

[12]Available at https://epic.org/privacy/cookies/flash.html
[13]Available at https://html.spec.whatwg.org/

While `sessionStorage` is only stored during session time and, therefore, has no useful application for fingerprinting, `localStorage`, on the other hand, is designed for storage that spans multiple windows, and persists after the browser is closed.

Both, (Acar et al., 2014) and (Roesner et al., 2012) refer to the use of the `localStorage` mechanism as a way to perform browser fingerprinting.

Advantages: HTML's `localStorage` might be a concept somewhat obscure to most web users. The functionalities of history, HTTP cookies and (normal) cached content cleaning, available in most browsers nowadays, might trick users into thinking that, once used, all browsing content related data will be successfully wiped from the system. Until browsers start alerting the users of this data storing and provide a simple mechanism to manage this type of data, users will be exposed to the possibility of having a persistent ID etched to their browser.
Disadvantages: In their nature, Flash's Local Shared Objects and HTML's `localStorage` are cookies. This means that the use of such mechanisms falls under the Article 5(3) of Directive 2002/58/EC, amended by the Directive 2009/136/EC, which requires prior informed consent for storage or access to information stored on a user's terminal equipment. In other words, websites using Flash or HTML5 cookies must ask users if they agree with the storing of data before the site starts to use them, risking penalties when not abiding to these obligations.

4.9 Taxonomy

The taxonomy we present in Table 15 classifies each technique according to the type of data that is collected. Whenever possible, categories were created to group techniques according to the source of device-related data that they explore (leftmost column). Additional technique-related information is shown, such as, the type of fingerprinting performed (browser, cross-browser or both), if there is information written to the client's system (Active or Passive), and whether the techniques rely on a comparison database to perform the fingerprinting.

It should be noted that the Flash retrieval of OS features also comprises data about the Flash plugin itself, therefore, providing information about the browser (browser and cross-browser fingerprinting). HTTP Header fields also allow both types of fingerprinting. This happens because properties, such as the "User-Agent" field, provide both browser and system information.

Table 15: Taxonomy of the web-based fingerprinting techniques.

Technique		**Browser fingerprinting**	**Cross-browser fingerprinting**	**Active/Passive**	**Requires comparison database**
IP address		NO	YES	PASSIVE	NO
HTTP header fields		YES	YES	PASSIVE	NO
Browser properties	**Plugin enum.**	YES	NO	PASSIVE	NO
	Do-Not-Track	YES	NO	PASSIVE	NO
Browser behaviour	**Rounding pixels**	YES	NO	PASSIVE	YES
	Canvas	YES	NO	PASSIVE	YES
	Performance	YES	NO	PASSIVE	YES
Operating system features	**Flash plugin**	YES	YES	PASSIVE	NO
	Java plugin	NO	YES	PASSIVE	NO
	Clock skew	NO	YES	PASSIVE	YES
Hardware features	**Screen properties**	NO	YES	PASSIVE	NO
	CPU and RAM	NO	YES	PASSIVE	NO
	Network interfaces	NO	YES	PASSIVE	NO
Font enumeration		NO	YES	PASSIVE	YES
Cached objects	**HTTP cache**	YES	NO	ACTIVE	NO
	History	YES	NO	ACTIVE	NO
	Flash cookie	YES	NO	ACTIVE	NO
	HTML5 cookie	YES	NO	ACTIVE	NO

5 CONCLUSION

Due to its almost completely unnoticeable nature, web-based fingerprinting raises several privacy issues. The obvious one is the possibility to track a user's online behaviour without his consent. Web-based fingerprinting also poses major risks to security: by collecting software versions (of browsers and plugins) and operating system releases, a *fingerprinter* can gather enough information about a system to perform a successful attack. This was the basic premise of the *Blackhole Exploit Kit* (BHEK) - an exploit kit, created in 2010, designed to identify software vulnerabilities in client machines communicating with it and exploiting discovered vulnerabilities to upload and execute malicious code on the client. The developer of computer security software, Sophos, published a report (Howard, 2012) (p. 9) about BHEK where it states that "[t]he purpose of the landing page is straightforward: to fingerprint the machine. The landing page used by Blackhole uses code from the legitimate PluginDetect library to identify: OS, Browser (and browser version), Adobe Flash version, Adobe Reader version, Java version".

Sadly, the mitigation measures that exist today fall short for what is needed. (Nikiforakis et al., 2015) propose a "solution to the problem of browser-based fingerprinting", by modifying the browser to make every visit appear different to a fingerprinting site. However, this solution addresses only JavaScript based techniques, like font/plugin enumeration and screen resolution and, as shown in the current work, there is a myriad of other techniques that PriVaricator does not address, rendering it still fingerprintable. Tor Browser, another proposal for countering fingerprinting, has several limitations and constraints. The use of Tor networks adds specific constraints (e.g. longer connection delays, script blocking) and risks (e.g. confidentiality at the exit node), which would render the benefits of its use pointless.

The work we present in this paper depicted a study of the currently known web-based fingerprinting techniques. A special effort was put into gathering the most complete universe of techniques. Techniques were categorized in a taxonomy that will allow readers to easily identify different types of techniques, their capabilities, limitations and similarities. The categories were defined bearing in mind that new techniques might appear in the future and still have a category where they will fit in.

Finally, this study provides a working basis for future research in the field of web-based fingerprinting mitigations by presenting a complete and understandable insight of the currently known techniques.

The Web-based fingerprinting concept emerged after the realization that a device becomes unique because it is used by a unique entity-in this case, a human being. As devices provide more customizations to match the users preferences, the more individualized they become, increasing the risk of fingerprinting.

ACKNOWLEDGEMENTS

This work is partially supported by National Funding from FCT - Fundação para a Ciência e a Tecnologia, under the project UID/CEC/00408/2013.

REFERENCES

Acar, G., Eubank, C., Englehardt, S., Juarez, M., Narayanan, A., and Diaz, C. (2014). The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 674–689. ACM.

Acar, G., Juarez, M., Nikiforakis, N., Diaz, C., Gürses, S., Piessens, F., and Preneel, B. (2013). Fpdetective: dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1129–1140. ACM.

Article 29 Data Protection Working Party, A. (2014). Opinion 9/2014 on the application of directive 2002/58/ec to device fingerprinting.

Boda, K., Földes, Á. M., Gulyás, G. G., and Imre, S. (2012). User tracking on the web via cross-browser fingerprinting. In *Information Security Technology for Applications*, pages 31–46. Springer.

Eckersley, P. (2010). How unique is your web browser? In *Privacy Enhancing Technologies*, pages 1–18. Springer.

Fielding, R. and Reschke, J. (2014). Hypertext transfer protocol (http/1.1): Semantics and content.

Howard, F. (2012). Exploring the blackhole exploit kit. *Sophos Technical Paper*.

Janc, A. and Olejnik, L. (2010). Web browser history detection as a real-world privacy threat. In *Computer Security–ESORICS 2010*, pages 215–231. Springer.

Jenkins, I. R., Shapiro, R., Bratus, S., Speers, R., and Goodspeed, T. (2014). Fingerprinting IEEE 802.15.4 Devices with Commodity Radios. Technical Report TR2014-746, Dartmouth College, Computer Science, Hanover, NH.

Khademi, A. F., Zulkernine, M., and Weldemariam, K. (2015). An empirical evaluation of web-based fingerprinting. *Software, IEEE*, 32(4):46–52.

Kohno, T., Broido, A., and Claffy, K. C. (2005). Remote physical device fingerprinting. *Dependable and Secure Computing, IEEE Transactions on*, 2(2):93–108.

Mayer, J. R. (2009). Any person... a pamphleteer: Internet anonymity in the age of web 2.0. *Undergraduate Senior Thesis, Princeton University*.

Mayer, J. R. and Mitchell, J. C. (2012). Third-party web tracking: Policy and technology. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 413–427. IEEE.

Mowery, K., Bogenreif, D., Yilek, S., and Shacham, H. (2011). Fingerprinting information in javascript implementations. *Proceedings of W2SP*, 2.

Mowery, K. and Shacham, H. (2012). Pixel perfect: Fingerprinting canvas in html5. *Proceedings of W2SP*.

Nikiforakis, N., Joosen, W., and Livshits, B. (2015). Privaricator: Deceiving fingerprinters with little white lies. In *Proceedings of the 24th International Conference on World Wide Web*, pages 820–830. International World Wide Web Conferences Steering Committee.

Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F., and Vigna, G. (2013). Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Security and privacy (SP), 2013 IEEE symposium on*, pages 541–555. IEEE.

Olejnik, L., Castelluccia, C., and Janc, A. (2012). Why johnny can't browse in peace: On the uniqueness of web browsing history patterns. In *5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2012)*.

Roesner, F., Kohno, T., and Wetherall, D. (2012). Detecting and defending against third-party tracking on the web. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 12–12. USENIX Association.

Zalewski, M. (2012). *The Tangled Web: A Guide to Securing Modern Web Applications*. No Starch Press.