Analysis of the use
of
Obfuscated Web Tracking

Author:
Federico Fallace

Supervisor:
Pere Barlet-Ros
Co-Supervisor:
Marco Mellia

June 2017
Indice

1 Introduction 7
   1.1 Web Tracking ........................................ 7
   1.2 Fingerprinting ........................................ 10
   1.3 Canvas Fingerprinting ................................. 11
   1.4 Obfuscated Programming ............................... 15
   1.5 Report Structure ..................................... 16

2 State of the art 19
   2.1 Measurement ........................................... 19
   2.2 Countermeasures ..................................... 21
      2.2.1 GHOSTERY and DoNotTrackMe .................... 22
      2.2.2 Tor .............................................. 22
      2.2.3 Firegloves ...................................... 23
      2.2.4 Do Not Track .................................... 23

3 Scope of the project 24
   3.1 Objectives ............................................ 24
   3.2 Scope ................................................. 24
3.3 Useful courses ........................................... 25
3.4 Competences ........................................... 26
3.5 Stakeholders ........................................... 26

4 Project Planning ........................................... 27
   4.1 Task Description ....................................... 27
       4.1.1 Possible deviations and alternatives ............. 31
       4.1.2 Action plan ....................................... 31
       4.1.3 Gantt chart ....................................... 32
   4.2 Tools and resources .................................... 33
       4.2.1 Hardware ......................................... 33
       4.2.2 Software ......................................... 33
   4.3 Budget Analysis ....................................... 33
       4.3.1 Human Resources ................................ 33
       4.3.2 Hardware costs .................................. 34
       4.3.3 Software costs .................................. 34
       4.3.4 Total expected cost ................................ 34
   4.4 Sustainability analysis ................................ 35
       4.4.1 Environmental impact ....................... 35
       4.4.2 Social impact ..................................... 36
       4.4.3 Economic impact ................................ 36
       4.4.4 Sustainability matrix ....................... 36

5 Methodology ........................................... 37
   5.1 Scraper ............................................... 39
       5.1.1 Main library used ............................. 40
INDICE

5.2 Mozilla Firefox Plugin .................................. 41
5.3 Server .................................................. 43
5.4 Database .................................................. 45

6 Results and conclusions .................................. 48

7 Future works ............................................. 57

Indice
Abstract

In the last years, web tracking has become a fast-growing phenomenon. Profiling users to provide targeted advertisement is a business that counts hundreds of companies and billions of dollars. On the other hand, communities, researchers and other companies are building countermeasures to prevent tracking practices, so the techniques are becoming more sophisticated and hidden. This work has the goal of uncovering the obfuscation that is becoming common in web tracking methods and, in particular a popular tracking method called canvas fingerprinting. The proposed approach could also be used in the future for other tracking techniques. Our tests seek also to uncover web tracking methods not situated in the home pages, but in the sub links, in order to discover if there is a substantial difference. We crawled more than 830K links presents in the home pages of the first 5K most visited web sites according to Alexa’s ranking. Our tool uncovered the real calls of the canvas fingerprinting method toDataURL(), making it impossible to hide by web trackers. The results showed that 12% of the analyzed domains have plain-text canvas fingerprinting methods in the home page, while 1,2% uses obfuscation and 86,8% is canvas free. On the other hand, when we analyzed the sub links, the percentage increased to 30,5% for plain-text canvas.
fingerprinting and to 10.5% for the obfuscated one, while only 59% of the domains were canvas free. In addition, we uncovered 2695 trackers and just the 3 most popular covered more than 20% of the visited domains. Finally we analyzed the files from where the tracking method was called, and we found out that the same tracking code is used in many different domains; the most widespread was tanxssp.js, present in 71 different domains.
Capitolo 1

Introduction

1.1 Web Tracking

Nowadays Internet has become an essential part of our lives and our daily actions. Shopping, staying in contact with friends, working, searching information about our hobbies and our travels are only some examples of how we are more and more connected in each single aspect of our life.

On the other hand, web advertisement is constantly growing and, according to [1], its revenues have surpassed TV broadcast revenues since 2005, because they are cheaper and more targeted. Since every day billions of users put sensitive information in the web, it is not difficult to understand that the business about users’ tracking is very lucrative and fast-growing. Most web services are collecting information about users, and more specifically about their searches, visited web sites, contacted people, bought products and more. Although this information is gathered for commercial purposes, the ways of usage are far different from the simple targeted advertising. Some recent
studies [2] indeed, have shown that purposes include also price discrimination, health and mental condition [3, 4] and financial reliability assessment [5, 6, 7].

In the last years web privacy measurements detected, described and quantified services with privacy-impacting behaviours, forcing companies to improve privacy practices, to answer to public pressure, regulatory actions and press coverage [8, 9].

One of the most concerning aspects about web tracking is that users can give information with their will, for instance filling a web form or accepting the transfer of specific information. On the contrary most of the time data collection is done without users’ knowledge. In particular users do not know the methodologies web trackers use to take information, neither which specific information is taken.

The information that is usually collected can be both sensitive and technical. For the former type there are the geographical location, the preferences or even the history of visited web pages, while for technical information we can find the used browser, the operating system, the IP address, the used hardware and so on.

The methodologies used by web trackers are several, for instance analysis of the IP address, HTTP Requests or also programs and scripts in Flash and JavaScript. In the last category there is canvas fingerprinting, the methodology analyzed in this thesis. In the very few last years, some studies have described the mechanisms used to track users [10] and have done huge tests on the most visited websites [11]. The used methodologies are always
fast-growing and for this reason online tracking has been described as an "arms race". Indeed mechanisms are becoming really difficult to detect, to control and also to delete. Nowadays it is almost impossible to cancel all information about you and start with a new and clean profile. With some tools it is possible to block part of the tracking, but often they cause losses in content or functionalities.

In 1994 cookies were introduced in the context of web browser by Lou Montulli [12]. It was a big innovation for web developers and browser vendors because it transformed in state-full the HTTP protocol, that is state-less on its own. The basic concept of cookies is that the server can save a few data in the browser and then send them back with subsequent requests. Not so much time after their introduction, some abuses were observed. Indeed one web page can have different files which can be located in different servers (obviously the one hosting the main page, but also third-party ones) and all of them are able to create their own cookies. So if the same server can create cookies on a lot of website, it can track the user through the websites and create his browsing history. This phenomenon is called third-party cookies.

Soon the community answered with countermeasures:

- a discrete part of users started to delete both first and third-party cookies once a month;

- tools to detect the tracking were created (for instance Ghostery);

- browsers developed already built-in options to avoid third-party cookies;
browsers created private mode, that avoid to leave traces of the visits on the devices.

Advertisers and trackers had to develop some other ways to track users and in 2010 with Eckersley’s work [13], it was clear how to identify devices and users without using cookies.

1.2 Fingerprinting

In the last years, the browser has become the main tool for choices in Internet; it chooses the websites and the users to trust, and it gives a correct visualization of the online services. To perform these operations it has to give some information about installed software and used hardware to web services, that will be able to correctly render contents or to serve device-compatible media. In order to execute efficiently the set tasks, the browser is always more tied to Operating System functionalities and the system’s hardware, and consequently websites’ programmers have more access to the resources. The problem is that the APIs, usually used to ask resources’ information for the correct visualization, are flexible enough to be used to define a fingerprint, unique (or almost) for each device. This practice is called web-based device fingerprinting and it has worrying privacy and security implications.

We can define the Eckersley experiment in 2010 [13] as the official discovery of the fingerprint. He supposed that information like screen dimensions, installed fonts and so on, could be combined to create a device-specific fingerprint. Different attributes were used with different priorities depending on how much they are common between users and how much they are stable.
in a device. The results of the experiment showed that 94.2% of the devices had a unique fingerprint. These results are limited to devices using JavaScript and Flash, but they are still worrying if we think that users can be identified and tracked without stateful client-side technologies (like cookies). In this way, they are able to track users also if they avoid the use of browser’s or Flash cookies, circumventing users’ preferences about tracking and limitations imposed by Europe and United States regulations.

In Figure 1.1 they are shown the properties that the browser is able to detect and an example of fingerprint of the computer used. The picture is just a snapshot of an experiment you can repeat on the website [14]. Fingerprint can be used to unify users’ data collected from different devices in a unique profile. The information collection works with databases, where a device is added if it is unknown, or matches with a profile if it’s previously known. The purposes can be positive for the users, for instance anti-fraud systems, but also against their interests and wills, as in the case of tracking and advertisement.

1.3 Canvas Fingerprinting

Canvas fingerprinting is the most common fingerprinting method ever studied and it was presented for the first time in the paper [15]. With HTML5 the new <canvas> element was introduced, which provides an area of the screen where it is possible to draw. It is compatible with most recent versions of Chrome, Firefox, Safari, Internet Explorer, Opera and also mobile Safari and Android Browser. Using HTML tag <canvas> and its APIs, it is possible
Canvas Support in Your Browser:
- Canvas (basic support): True
- Text APIs for Canvas: True
- Canvas.toDataURL: True

Database Summary:
- Unique User-Agents: 176794
- Unique Fingerprints: 6213

Your Fingerprint:
- Signature: DA85E084
- Uniqueness: 99.94% (102 of 176794 user agents have the same signature)

Image File Details:
- File Size: 6627 bytes
- Number of Colors: 998
- PNG Hash: 190EE45839D0C8FB50612E6D6D929472
- PNG Headers:

<table>
<thead>
<tr>
<th>Chunk</th>
<th>Length</th>
<th>CRC</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>IHDR</td>
<td>13</td>
<td>477A703E</td>
<td>PNG image header: 220x30, 8 bits/sample, truecolor+alpha, noninterlaced</td>
</tr>
<tr>
<td>sRGB</td>
<td>1</td>
<td>A6CE1CE9</td>
<td>sRGB color space, rendering intent: Perceptual</td>
</tr>
<tr>
<td>IDAT</td>
<td>6557</td>
<td>DA85E084</td>
<td>PNG image data</td>
</tr>
<tr>
<td>IEND</td>
<td>0</td>
<td>AE426082</td>
<td>end-of-image marker</td>
</tr>
</tbody>
</table>

Browser Statistics:
Looking at your signature, it’s very likely that your web-browser is Safari and your operating system is Mac OS X.

Operating Systems:
- Mac OS X: 102/102
- OS by Version:
  - Mac OS X 10.11: 67/102
  - Mac OS X 10.12: 35/102

Browsers:
- Safari: 82/102
- Apple Mail: 20/102

Devices:
- Other: 102/102

Platforms:
- MacIntel: 102/102

Safari 10.0: 34/102
Safari 9.1: 22/102
Safari 9.0: 18/102
Safari 10.1: 6/102
Apple Mail 602.2: 4/102
Apple Mail 602.1: 4/102
Apple Mail 601.7: 4/102
Apple Mail 602.3: 2/102
Apple Mail 601.6: 2/102
Safari 8.1: 1/102

Figura 1.1: Information extractable from the browser
1.3. CANVAS FINGERPRINTING

Figura 1.2:

to detect small differences in the rendering of a text or a WebGL scene. In this way it is possible to obtain a fingerprint in very few time and without users’ knowledge.

The canvas element is just an area, but with the context 2d and its function `getContext()` we obtain an object that provides methods and properties for drawing on the canvas. Here we have a list of methods for this purpose by [16]:

In particular we focus our work on the call `toDataURL()`. This method returns a data URL consisting in the Base64 encoding of the PNG image containing the entire contents of the canvas area.

In the Figure 1.2 from [11] we can see the basic functioning to fingerprint canvas. On the website visit, the script draws text with particular font, size and background with a script similar to the one in Figure 1.3. Then `toDataURL()` is called to get the image of the canvas element in the Base64 encoding. Finally the script hashes it and the fingerprint is obtained. This method can also be combined with other browser properties as list of plu-
Colors, styles and shadows properties
fillStyle, strokeStyle, shadowColor, shadowBlur, shadowOffsetX, shadowOffsetY

Colors, styles and shadows methods
createLinearGradient(), createPattern(), createRadialGradient(), addColorStop()

Line styles properties
lineCap, lineJoin, lineWidth, miterLimit

Rectangles methods
rect(), fillRect(), strokeRect(), clearRect()

Paths methods
fill(), stroke(), beginPath(), moveTo(),.closePath(), lineTo(), arc(), arcTo(), isPointInPath()

Transformations
scale(), rotate(), translate(), transform(), setTransform()

Text properties
font, textAlign, textBaseline

Text methods
fillText(), strokeText(), measureText()

Image Drawing
drawImage()

Pixel Manipulation properties
width, height, data

Pixel Manipulation methods
createImageData(), getImageData(), putImageData()

Compositing Methods
globalAlpha, globalCompositeOperation

Other methods
save(), restore(), createEvent(), getContext(), toDataURL()
gins, fonts or the user agent string. Different operating system, font library, graphics card, graphics driver and the browser differentiate the rendering of the canvas element. On the website [14] it is possible to see an example of your device fingerprint.

1.4 Obfuscated Programming

With Obfuscated Programming we mean a transformation of the code that makes the code more difficult to read, to understand and to change. More difficult in terms of needed human resources, computational power and money required to fully understand it.

The purposes are several, for instance avoiding code theft and reuse by competitors or in general programmers, protecting intellectual property, adding a security layer. On the other hand obfuscation can be used also to hide malicious code, like in the web tracking case we analyzed. This methodology is used because in some cases delivering the source code is mandatory or just the best design choice. Some example: a server is not available or is too expensive, mobile applications or offline games. In these and several other cases, there is the strong need to protect your code.

A common misunderstanding is confusing obfuscation with encryption, although these two concepts are really different. The former is still executable, and it does not need a function to be deobfuscated; the encrypted code is not ready to be executed, it needs a decryption before.

An other misunderstanding is to confuse minification with obfuscation. These two concepts share often the same techniques, but the goals are different.
The former is used to compress the code in order to make it smaller and faster, especially if we are talking about web services.

The techniques used to obfuscate the code are several. In Figure 1.4 we can see the original version of a sample function, while in Figure 1.5 there is the same code after we applied the Renaming of variables and functions and the Comment removal. In Figure 1.6 we also applied Whitespace removal, while in Figure 1.7 the String splitting. Other common techniques are the Dead code injection and Non alphanumeric Obfuscation.

1.5 Report Structure

This paper will describe the project and the analyzed problem in the following sections. In Chapter 3 we have a description of the main previous
1.5. REPORT STRUCTURE

contribution in this field, so its State of the art. The Scope of the project and the needed competencies to develop it are described in Chapter 4. In Chapter 5 it is analyzed the Sustainability of the project, in terms of cost, human resources and time. In Chapter 6 there is a detailed description of the Methodology used to develop the tool of this work. Their Results and the conclusions we have extracted are in Chapter 7. Finally, in Chapter 8 there are hints for Future researches.
Figura 1.7: Example of string splitting
Capitolo 2

State of the art

There are multiple tools to measure and counteract web tracking in the wild. In the next two subsections they will be described measurement works, studies describing web tracking and more specifically fingerprinting, and finally some of the countermeasures present in the wild.

2.1 Measurement

We started our project continuing the work of a Master Thesis from Alvaro Espuna Buxo’ and the same supervisor Pere Barlet-Ros [17]. This work uncovered the obfuscated web tracking but limited the analysis to the first 10K most visited websites according to Alexa’s ranking. In the Survey [10] by Bujlow and Barlet-Ros, supervisor of this thesis, we find a comprehensive description of the whole literature in the field of web tracking methods, but also their purposes, implications and possible users’ defenses. They divide tracking mechanisms in 5 categories: Session-only, Storage-based, Cache-
based, Fingerprinting and Other ones. In our work we focused on the 4th category.

The 2010’s work [13] by Eckersley is the first published study on fingerprinting, and it deeply describes several fingerprinting techniques and which device properties are better to have unique fingerprints.

In the paper [12] by Nikiforakis, we can find an other good analysis on how web-based device fingerprinting works, with also the explanation of how and why this tracking mechanism was born. Finally, the research paper [15] describes the canvas-based tracking techniques more in details.

In the last years several works have measured the presence and invasiveness of web tracking in the modern Internet.

For sure the largest of them is [18] by Englehardt and Narayanan, that has measured different kinds of tracking methods in the top 1-million websites. They used the famous tool OpenWPM to implement an extensive analysis on 15 methodologies, including stateful and stateless tracking, the effects of browser’s privacy tools, and the exchange of tracking data between different web sites. It is notable also their previous work [11] with Acar, that focuses on canvas fingerprinting, evercookies and its conjucted use of cookie syncing.

Always by this last author, the work [19] presents a new tool, FPDetective, to detect the fingerprinting itself, without the use of balcklists of known web trackers.

To conclude the list of main paper about web tracking measurement, we have the notable work by Metwalley, Traverso and Mellia [20], that focuses on the detection of users’ identifiers and that uncovered 34 new third-party trackers not present in previous blacklists. Their other paper [21] is also notable and,
in our opinion, differentiate a lot from previous works because of the analyzed datasets, made up of real users’ navigations data, obtained from 2 ISPs.

The main points that differentiate our work from these previous ones are the following: firstly, our tool executes a dynamic analysis of the code on the actual JavaScript calls, so without static pattern-matching; in this way, we are able to detect obfuscated web tracking, that, in our hypothesis, is spreading in modern websites to not be uncovered by existing tools.

The second difference is in our web crawler. While previous works focused on the home pages of visited web sites, we went deeper, on the second layer domain links. Indeed we supposed that canvas tracking methods and obfuscation could be more present in pages different from landing ones. The reasons are several: useful information about our interests, our searched objects and so on, is more likely to be exposed in sub pages than on the landing pages. In addition, the presence of web tracking on 2nd or 3rd level domain links is still unknown and could also have been moved there as a consequence of the results from previous works.

2.2 Countermeasures

Preventing device fingerprinting is really difficult, but there are already some methodologies, more or less efficient, that are trying to avoid it. In the next lines we present the main ones.
2.2.1 GHOSTERY and DoNotTrackMe

These two tools are commercial anti-tracking extensions for browsers. We have some concerns about these tools:

- they have low usage percentages; from [20] we know that around 12% of users actually installed them;
- they block only partially the information sent to trackers;
- they rely on blacklists, built online and periodically updated (once per day or each bootstrap of the browser), but we do not know how these list are built.

Most of the people do not know or do not care about tracking and these tools. They are more interested in deleting advertisement from their online life.

2.2.2 Tor

The Tor browser is the basic tool to access the Tor anonymity network, a service that daily allows 800k people to browse completely anonymously. In their privacy requirements, there is the cross-origin fingerprinting unlinkability. From this premise, it is obvious that it incorporates strong defenses to fingerprinting. From the test made by [19], although most of fingerprint attributes (especially the browser-related ones) were uniformed so impossible to be used, there were some leaks on the fonts list. They were fixed with the next update, but the community has to be always aware of the new updates to continously prevent leak of information.
2.2. COUNTERMEASURES

2.2.3 Firegloves

Firegloves is a Mozilla Firefox extension, born for research purposes. Once installed, the browser answers to requests about screen resolution, running platform, browser version and so on, with randomized information. From the tests [19] in 2014, there were some ways to avoid this protection. Using different APIs or Flash, it was still possible to know information like dimension of the text (used in font-based fingerprinting), the used Operative System and so on. Additionally, it was possible to understand if this extension was in use and, since less than 2000 people were using the tool, it was a high priority attribute to build their fingerprints, becoming counter-productive.

2.2.4 Do Not Track

Do Not Track (DNT) is a HTTP header field currently standardized by the W3C and used in the most famous web browsers. It basically allows users to express preferences on being tracked or not. The problem is that it is only a request that can be heard or not. From the test of [19] we can conclude that none, or at least a minimum part of trackers, considers the users’ preference.
Capitolo 3

Scope of the project

3.1 Objectives

The main objectives of this project are the following. We want to uncover:

- how much canvas fingerprinting is used in the modern websites;

- if and how much obfuscated programming is used by web trackers;

- if there is a substantial difference between tracking in the landing pages and in the links present in the home pages.

3.2 Scope

The scope of this work is research-driven: we want just to answer to the questions presented in the previous paragraph. We execute the tests with our tool, to give answers to our questions. If they will be different from our
3.3. USEFUL COURSES

hypotheses, the work will still be useful, because it would add previously unknown information to literature.

3.3 Useful courses

**APA, Ambient Intelligence and Software Engineering**
With these courses, I have learned advanced programming, Python language and how to manage a project.

**Distributed Programming**
This course was useful for the basic knowledge about HTML, JavaScript and the web services' functioning.

"Database" and "Database management system"
The basic knowledge about database was essential. I have combined it with the Python programming to create and manage the database. In addition, this knowledge was used to extract information from the tests' results.

**TMIRI**
Thanks to this course, I was able to discover efficient tools to find good references and, more important, proper methods to write in the scientific field.
3.4 Competences

Main programming languages: Python, SQLite, JavaScript.

3.5 Stakeholders

In this project we have 4 main stakeholders:

Developer and author
The person who implemented the system and the chosen methodology, and wrote this thesis, describing the project and its results.

Project supervisor
The project supervisor is Pere Barlet-Ros. His function is to guide and help the developer on critical points and analysis of the results.

Scientific and Open Source communities
They provide research studies, libraries and useful tools that were essential for this project.

Target audience
It is both the research community and the average Internet user. The objective of this research is to uncover part of web tracking, and consequently to raise awareness about its ubiquity.
Capitolo 4

Project Planning

4.1 Task Description

After an initial and general planning, all the smaller parts of the project were planned, developed, tested and then planned again and so on, to insert missing parts not considered in the beginning. So the tasks 1,3,4,5 were not done in an unique block, but in small cycles, using a technique similar to Scrum. If the tests had been done in the end of the project, modifications in order to obtain a lot of missing information that we needed, or just to make it more efficient, would have required more time or they could have been useless.

The main tasks of the project were:

1. Reading scientific articles and study about the topic
2. Initial planning
3. Planning of the small tasks
4. Code

5. Test code

6. Running and supervise tests

7. Results’ evaluation

8. Report writing

9. Report revision

10. Oral defense preparation

Reading scientific articles

The topic of web tracking is not so common in the average career of Computer Engineering, so a deep study about it, and about all the other fields used in the projects, was mandatory. Additionally, a good knowledge about previous works was useful to direct this project to right choices.

<table>
<thead>
<tr>
<th>Tabella 4.1: Reading scientific articles</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expected duration</strong></td>
</tr>
<tr>
<td><strong>Human Resources</strong></td>
</tr>
<tr>
<td><strong>Material Resources</strong></td>
</tr>
<tr>
<td><strong>Task dependencies</strong></td>
</tr>
</tbody>
</table>

Initial planning

After reading up on the topic, we focused and identified the goals of the project and how to reach them.
4.1. TASK DESCRIPTION

Tabella 4.2: Initial planning

<table>
<thead>
<tr>
<th>Expected duration</th>
<th>30 hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resources</td>
<td>Thesis Author and Project Supervisor</td>
</tr>
<tr>
<td>Material Resources</td>
<td>Office computer</td>
</tr>
<tr>
<td>Task dependencies</td>
<td>None</td>
</tr>
</tbody>
</table>

Planning of the tasks

After a general planning, each task was isolated and a developing solution was thought, using the algorithm design paradigm "Divide et impera".

Tabella 4.3: Planning of the tasks

<table>
<thead>
<tr>
<th>Expected duration</th>
<th>70 hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resources</td>
<td>Thesis Author</td>
</tr>
<tr>
<td>Material Resources</td>
<td>Office computer</td>
</tr>
<tr>
<td>Task dependencies</td>
<td>Initial planning</td>
</tr>
</tbody>
</table>

Code

This task refers to the real implementation of the code.

Tabella 4.4: Code

<table>
<thead>
<tr>
<th>Expected duration</th>
<th>120 hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resources</td>
<td>Thesis Author</td>
</tr>
<tr>
<td>Material Resources</td>
<td>Office computer</td>
</tr>
<tr>
<td>Task dependencies</td>
<td>Planning of the tasks</td>
</tr>
</tbody>
</table>

Test code

In this part we tested the correct working of the written code.
Running and supervise tests

This task consists in running the code to obtain data we need. Since the tests are large, a strong supervision was needed.

Results’ evaluation

Once we had the results, we were able to evaluate them and confirm our hypothesis.

Report writing and revision

In parallel with the execution of the tests, it was needed to write this report to explain our hypothesis, our methodology and our results.
4.1. TASK DESCRIPTION

Tabella 4.8: Report writing and revision

<table>
<thead>
<tr>
<th>Expected duration</th>
<th>180 hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resources</td>
<td>Thesis Author and Project Supervisor</td>
</tr>
<tr>
<td>Material Resources</td>
<td>Office computer</td>
</tr>
<tr>
<td>Task dependencies</td>
<td>Planning</td>
</tr>
</tbody>
</table>

Oral defense preparation

In the end the project has to be presented to a commission, so in this part there was the preparation of the presentation and the oral defense.

Tabella 4.9: Oral defense preparation

<table>
<thead>
<tr>
<th>Expected duration</th>
<th>40 hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resources</td>
<td>Thesis Author and Project Supervisor</td>
</tr>
<tr>
<td>Material Resources</td>
<td>Office computer</td>
</tr>
<tr>
<td>Task dependencies</td>
<td>Report writing and revision</td>
</tr>
</tbody>
</table>

4.1.1 Possible deviations and alternatives

This project is fundamentally a research project and some deviations can occur as a consequence of the nature of the project. This should not create an alarming situation as long as the deviations are controlled and can fit in the project schedule. Therefore, it is very important to identify deviations and monitor them closely. For this reason, weekly meetings will be crucial.

4.1.2 Action plan

As the project is being done by one developer there is not a need for coordinating different people/teams. This means it is possible to revise and adapt dynamically the initial planning. If one of the phases is longer than expected
the inevitable consequence will be that the remaining phases will be shortened in time. As stated before, some deviations can occur and it will be crucial to address them as part of the weekly progress assessment. As a last resort, if one of the phases were to take too long to accommodate in the timeline, initial requirements will need to be simplified.

4.1.3 Gantt chart

![Figura 4.1: Gantt chart part I](image)

![Figura 4.2: Gantt chart part II](image)
4.2 Tools and resources

4.2.1 Hardware

For the project it was used a computer in the office, with the following characteristics:

- OptiPlex 7010 by Dell Inc. 64 bits
- Intel(R) Core(TM) i5-3470 CPU @ 3.20 GHz
- RAM 8GB

4.2.2 Software

The used operative system was Ubuntu 16.04.2 LTS.
The used programming languages were Python, Javascript and SQLite.
The main libraries were Selenium, bottle, html, sqlalchemy, urlparse3 and tlsh.
Other used tools were Overleaf, Google Drive and TeamViewer.

4.3 Budget Analysis

4.3.1 Human Resources

The Human Resources needed for this project are:

- Project Director
- Software designer
CAPITOLO 4. PROJECT PLANNING

- Software programmer

- Software tester

The needed hours and the costs are summed up in the table below.

<table>
<thead>
<tr>
<th>Role</th>
<th>Estimated hours</th>
<th>Price per hour</th>
<th>Total cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervisor</td>
<td>40</td>
<td>60</td>
<td>3600</td>
</tr>
<tr>
<td>Software designer</td>
<td>350</td>
<td>30</td>
<td>10500</td>
</tr>
<tr>
<td>Software programmer</td>
<td>190</td>
<td>30</td>
<td>5700</td>
</tr>
<tr>
<td>Software tester</td>
<td>50</td>
<td>30</td>
<td>1500</td>
</tr>
<tr>
<td>Total</td>
<td>650</td>
<td>-</td>
<td>21300</td>
</tr>
</tbody>
</table>

4.3.2 Hardware costs

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Units</th>
<th>Useful life</th>
<th>Price/month</th>
<th>Amortization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office computer</td>
<td>1400</td>
<td>1</td>
<td>4 years</td>
<td>29.17€</td>
<td>145.85€</td>
</tr>
</tbody>
</table>

4.3.3 Software costs

The used software are described in Section 5.2.2. All of them are open source, so without additional costs.

4.3.4 Total expected cost

The costs about were explained and calculated in the paragraphs above, while the consumption is calculated in the next Section 5.4.1.
4.4 SUSTAINABILITY ANALYSIS

<table>
<thead>
<tr>
<th>Expense</th>
<th>Cost (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resources</td>
<td>21300</td>
</tr>
<tr>
<td>Hardware</td>
<td>145,85</td>
</tr>
<tr>
<td>Software</td>
<td>0</td>
</tr>
<tr>
<td>Consumption</td>
<td>20,4</td>
</tr>
<tr>
<td><strong>Total cost</strong></td>
<td><strong>21466,25</strong></td>
</tr>
</tbody>
</table>

4.4 Sustainability analysis

4.4.1 Environmental impact

In order to evaluate the environmental impact of the project, we calculated how many KWh were used, and consequently how much CO2 was emitted. The consumption of a middle-range computer (as the one used for the project) is about 150W per hour. The total computer working hours were around 1200. We calculated separately the screen consumption because, during most of the tests’ execution, the screen was not used. The screen has a consumption of 50W per hour, and the hours were around 480.

We applied the following expression to these numbers, to obtain the total consumed energy cost.

\[
\sum_{i=1}^{n} \frac{(\text{Device's consumption}[\text{W}] \times \text{number of hours})}{1000} = 204\text{KWh}
\]

In Spain the average cost of a KWh is 0,10 €, so the total cost for the energy is around 20,4€.

Finally, in according to [22], the average consumption in Spain for each KWh is 270g of CO2. So the total CO2 emitted for this project is **55,08 Kg**.

There is no manufacturing needed, and no waste is generated as a result of the project, nor as a result of its deployment or utilization. So the envi-
ronmental impact of the project is very low. Therefore, a high score on the environmental dimension is appropriate.

4.4.2 Social impact

Every day a huge amount of users is tracked while visiting websites. This work could be useful to be aware about it and take the possible countermeasures. So it can have a good social impact.

4.4.3 Economic impact

The total cost of the project was calculated in Section 5.3 (Budget Analysis), but since most of human resources were not payed because the project is a Master Thesis and the computer was unused property of the University, these costs are really low.

4.4.4 Sustainability matrix

<table>
<thead>
<tr>
<th>Category</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environmental</td>
<td>9/10</td>
</tr>
<tr>
<td>Social</td>
<td>8/10</td>
</tr>
<tr>
<td>Economic</td>
<td>9/10</td>
</tr>
<tr>
<td>Average</td>
<td>8,67/10</td>
</tr>
</tbody>
</table>
Capitolo 5

Methodology

In this chapter we describe before the general design of the project and then, in the next sections, we analyze the components in more technical detail. We started from the tool built in the project [17], then we modified and improved it.

Our system starts visiting the home pages and the links of first layer domain, namely the links present in the landing pages. In order to not make the tests’ execution too long and being able to crawl the first 10k websites by Alexa’s ranking, we did not crawl deeper.

For each page, we detect if there are actual calls of the canvas method `toDataURL()`, then we filter out the "legit calls" and finally we check if there was obfuscation of the code.

As legit calls, we mean the `toDataURL()` calls that are used to render better the canvas area, so with legit and not tracking purposes. To recognize them, we followed the constrain presented in [23], namely we did not consider canvas elements with properties `height` or `width` that are at least 16 pixels,
because they are unlikely to have tracking purpose. We remember that the default canvas size is 300x150px.

To check if the code is obfuscated, we look for the call (the string containing the function) in the original code, in particular in the line where toDataURL() was called, and also in the previous and the next line.

Now let’s see more technical details.

The project is made up by four main components:

- Scraper
- Mozilla plugin
- Server
- Database

Generally, we pass the .csv list of the most visited websites by Alexa’s ranking to the scraper, then it visit all web pages and, for each of them, it takes dynamically the links present in the HTML file and visit them. In
5.1 SCRAPER

the browser, in our case Mozilla Firefox, there is a plugin that replace the
JavaScript function toDataURL() with a personalized one. This injected
version does the same of the original function but sends also a json request to
the server, with some data like the canvas size, a snapshot of the document
in the moment of the execution and other that we will describe in the Mozilla
Firefox Plugin section. Finally the server processes the data and stores them
in a database. We can see the general design in the Figure 5.1, while in the
next sections all the components more in detail.

5.1 Scraper

The scraper is basically a web crawler. We pass to it as input a .csv file
with the list of the most visited websites from Alexa’s ranking and it opens
a Mozilla Firefox instance to visit them. Additionally, on every website it
visits, it downloads the HTML page and look for links, through the tag <a
href >. When it finds them it filters out links that for sure will cause a
TimeOutException (the browser is not able to load the page in the limit
time), for instance "#", "/", "None" and calls like "javascript:void(0);".
Then it add "http://", if not present, and change links from relative to
absolute. Finally it stores them in a list. All links present in each list are also
visited and the rest of the work is left to the plugin and the server. A counter
is incremented each time Firefox is not able to load the page, and finally this
value is substracted from the length of the list, so we have the exact number
of correctly loaded sub pages. The errors that can occur are the following:
LocationValueError, SSLError, TimeOutException, WebDriverException and
MaxRetryError. For each of them an exception is called, so the process is not blocked, but continues its execution. When all the sublinks of a website are visited, we store their number in a text file. Dynamically during the visits, if one of the exceptions listed above is called, the information about the error are also saved in a text file. In Figure 5.2 we can see the 5 main actions executed by the scraper, for each website present in the passed .csv file, in our case Alexa’s ranking.

5.1.1 Main library used

- Selenium
- Bs4
- Urllib3
- Requests
5.2 Mozilla Firefox Plugin

In this subsection we describe the Firefox plugin used in the system. The plugin basically injects a modified version of the JavaScript method `toDataURL()` in each visited webpage where this function is present. After the injection, it removes traces from the DOM, to try to reduce its footprint.

In this way, we will clearly take all the `toDataURL()` calls, comprehending also the legit ones, not used for tracking, but for correct visualization of the canvas content. We will see how we solved this problem.

calls the original function to have a similar execution and keep compatibility. Then raise an exception to be able to get a stacktrace. The plugin creates the injected function and then insert it as string in the part of the document where the original `toDataURL()` was called. The injected version does the following:

1. call the original function `toDataURL()`, to have a similar execution and maintain compatibility.

2. raises an exception, so we are able to get a stacktrace;

3. Saves in the variable param:

   - the canvas visible size with the properties scrollHeight and scrollWidth
   - a snapshot of the whole document in the moment of execution
   - the document.referrer, so the URL that loaded the document
   - the window.location.href, the URL of the current page
• the stacktrace

• the serialized subtree of the document

4. creates a json request and send the variable to the server with POST HTTP request method

5. removes the tree leaf of the script from the DOM.

From point 1 it is clear that the plugin has only a measurement scope, it does not block canvas fingerprinting.

Below the source code of the plugin:

```javascript
var scriptNode = document.createElement('script');

function instrument() {
    var old = HTMLCanvasElement.prototype.toDataURL;
    HTMLCanvasElement.prototype.toDataURL = function(c) {
        var trace = (new Error).stack;
        var xhr = new XMLHttpRequest();
        xhr.open("POST", our_server_address, true);
        xhr.setRequestHeader("Content-Type", "application/json");
        var params = {
            w: this.scrollWidth,
            h: this.scrollHeight,
            referrer: document.referrer,
            src: window.location.href,
            stack: trace,
        }
    xhr.send(JSON.stringify(params));
    HTMLCanvasElement.prototype.toDataURL = old;
    });
```

5.3. Server

This component is the core, where the gathered data are processed. It is written in Python language because it is simple but powerful, and it has a lot of useful libraries.

The server binds to an address and a port (to decide statically in the code) where the plugin will send the requests and it writes in a SQL database. Since we run everything on a single machine, the default address was the local host and we choose a random free port.

Since most of the tracking calls we are dealing with are by third parties we would like to know if the used files are the same. Also if we are dealing with
the same file, there will be some small differences, for instance the timestamp or user agent information, that are included dynamically in the response (we can see an example in Figure 5.3). We would hash these files and compare them more easily, without storing them completely, but a normal hashing function wouldn’t work. Indeed a function like one in SHA family would change the returned value also for a small difference, because actually this is the purpose of this kind of functions. So we used a locality sensitive hashing algorithm (in particular the library tlsh), to detect files’ equality, or better similarity.

An other important aspect is that the json request is not sent through the same domain, but from the visited page to our server. So we had to implement a valid response with the method OPTIONS, so it can be CORS compliant.
5.4 Database

The database schema below is quite simple, but it makes possible multiple queries at the end of the tests.

```
CREATE TABLE domain (
    id INTEGER NOT NULL,
    domain VARCHAR NOT NULL,
    alexa_rank INTEGER NOT NULL,
    PRIMARY KEY (id),
    UNIQUE (domain)
);
CREATE INDEX ix_domain_alexa_rank ON domain (alexa_rank);
CREATE TABLE log (
    id INTEGER NOT NULL,
    domain_id INTEGER NOT NULL,
    measured_at DATETIME,
    canvas_width INTEGER,
    canvas_height INTEGER,
    referrer VARCHAR,
    source_url VARCHAR,
    source_html VARCHAR,
    source_tlsh VARCHAR,
    stacktrace VARCHAR,
    st_caller_file VARCHAR,
    st_caller_line INTEGER,
);```
Here we explain the main fields:

**id/measured_at**
These two fields are an incremental id and the timestamp of the INSERT.

**domain_id**
This is the ranking of the domain of the URL that called the `toDataURL()` method.

**canvas_width/height**
Here we have the dimension of the canvas area.

**referrer/source_url**
This is the URL of the page where there was the call.

**source_html/source_tlsh**
This is the html page (transformed in a string) and its tlsh hash value.

**stacktrace**
This is the stacktrace that the exception of the injected version of the method called.

\texttt{st\_caller file/line/char/tlsh, st\_init file/line/char/tlsh}

Here we have the file and its tlsh hash value where toDataURL() was called. Additionally we have also the line and char of the call. The difference between the values st\_init and st\_caller is before and after the removing of the injected function.

\texttt{is\_obfuscated}

This is a boolean value that represents if there was or not obfuscation.
Results and conclusions

We analyze our results in order to discover if the following hypothesis were right:

- web tracking is becoming obfuscated
- the presence of web tracking on sub pages is bigger than in home pages.

These hypotheses are valid for web tracking, but we analyzed only the particular case of canvas fingerprinting. We crawled our tool on the first 5K websites of the most visited websites by Alexa’s ranking. We were able to reach 4209 of them, while on 3727 we found more than 0 links. The total number of links actually reached is 836653, while a bigger number of them were visited but not correctly loaded. In the following statistics we considered only the real numbers.

We will present before our results, divided for links, domains and trackers’ analysis, then we will discuss about them and we will extract conclusions about our hypothesis.
Links analysis

The total number of distinct URLs where we found plain-text canvas fingerprinting is 68836, while on 5974 we found obfuscated one. In the rest of the links, 761843, we did not find canvas fingerprinting tracking. In Figure 6.1 we can see these numbers in percentages. Since the number of links present

in each web site is really different (from 0 to 9983) this analysis does not give a real perception of the diffusion of this kind of tracking, but it is useful to understand the percentage of the obfuscation. From previous numbers, 8% of the total uncovered canvas fingerprinting is obfuscated, while 92% is in
Plain-text. We can see it in Figure 6.2.

Domains’ analysis

Our first test on domains where canvas fingerprinting is present, was done on the first 10K home pages, to observe potential differences between 2016 (last time the previous version of the tool was used) and 2017. The results (in Figure 6.3) showed that there was a decreasing of canvas fingerprinting in general, but there was a big increase of the obfuscation.

In our second and extensive test, we limited our analysis of the sub pages to the first 5K domains for time reasons, so we will compare their related results only to the first 5K home pages, although we have data until 10000th home page.

The domains that use canvas fingerprinting in their home pages are 13,2%, 12% in plain-text, while 1,2% with obfuscation. We found a big difference crawling also the sub pages: we found out that 41% of the domains are using canvas fingerprinting, 30,5% in plain text, 10,5% with obfuscation. We can see the difference in Figures 6.4 and 6.5.

Trackers’ analysis

Finally, we focused our analysis on the trackers. Since from the database we had the links of the files where toDataURL() was called, we had to build some small data analysis applications.

We firstly focused on detecting the trackers’ domains more widespread in the visited websites; this analysis was done to uncover the main third-party
trackers. We calculated on how many different websites domains each tracker
domain was present. For instance, the tracker domain doubleverify was pre-
sent on sina.com.cn, imgur.com and so on. In Figure 6.6 we can see the first
22 third-party trackers, the ones that were present in more than 10 websites.
In Figure 6.7 and 6.8 we divided the results for plain-text and obfuscated
canvas fingerprinting.

Then, we focused on the specific files used by the trackers, to see if the
same script is widely adopted in more websites. We can see the results
in Figure 6.9; the tanzssp.js was found on 71 domains, score.min.js on 45,
check.js on 21 and then other in fast decreasing.
Conclusions

The first hypothesis we want to confirm is that obfuscated programming is present in the web tracking, so all tools and measurements with static pattern matching analysis are not able to discover this part of the phenomenon. From figure 6.2 we can clearly understand that obfuscation is present and actually is a considerable part of canvas fingerprinting (8% of it) and since 10.5% of the visited domains use it, we can conclude it is also widely spread. In future studies it could be easily possible to extend this analysis to more tracking techniques using JavaScript, with just few modifications of our tool.

The second hypothesis we want to verify is that web tracking, in our study case only canvas fingerprinting, is more present in sub links than landing
Figura 6.6:

We supposed that 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 sub pages than on the landing pages. The second is that most of the previous works that analyze the presence of web tracking, focus only in the home pages, so the presence of web tracking on 2nd or 3rd level domain links is still unknown and could also have been moved as a consequence of the results from previous works.

From the data in Figure 6.4 and 6.5, we can see a clear difference. The presence of canvas on sub links is more than three times compared to the home pages (45.9% against 13.2%); if we analyze also these percentages divided in plain-text and obfuscated, we observe that the former is 2.5 times bigger, while the obfuscated canvas fingerprinting is almost 9 times bigger. So we
can conclude also that obfuscation is way more present in sub pages.

From the analysis of trackers we can also extract some other interesting data. We observe (in Figure 6.10) that the most widespread tracker, *doubleverify* covers more than 25% of the websites using canvas fingerprinting, while the second, *google*, around 12%, and the other ones in fast decreasing (atlassbx 6.18%, alicdn 4.18%, yimg 2.64% and so on). We can conclude that a big part of canvas fingerprinting is controlled by few trackers.
Figura 6.7:

Figura 6.8:
CAPITOLO 6. RESULTS AND CONCLUSIONS

Figura 6.9:

Figura 6.10:
Capitolo 7

Future works

In the future developing of our tool, we can consider technical and contents improvements.

For the former improvements it can be useful to decentralize the system, running the server on Internet or in a local network, and no more locally in the device. In this way it will be possible to run the scraper in parallel on many devices. It can be useful also to make the program lighter, for example using a multi-thread system architecture, and more efficient, creating some custom-made libraries.

These technical improvements can be implemented in order to have more extensive analysis, going deeper in the links and reaching the first 10K websites by Alexa’s ranking.

Additionally, with just some modifications of the Firefox plugin, it is possible to extend the analysis also to other JavaScript tracking techniques.
Bibliografía


[8] “Google Will Pay $22.5 Million to Settle FTC Charges it Misrepresented Privacy Assurances to Users of Apple’s Safari Internet Browser | Federal Trade Commission.”


