CANVAS FINGERPRINTING: A STATE OF THE ART

BY

AHMED ABOUOLLO

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

In

INFORMATION ASSURANCE AND SECURITY

November, 2018

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **AHMED ABOUOLLO** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN INFORMATION ASSURANCE AND SECURITY**.

Thesis Committee

Dr. Sami Zhioua (Adviser)

Dr. Moataz Ahmed (Member)

Sajjad Mahmood

Dr. Sajjad Mahmood (Member)

Dr. Hamoud Aljamaan
Department Chairman

Dr. Salam A. Zummo
Dean of Graduate Studies

12/2/19

Date

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

Dedication

This thesis is dedicated to my beloved parents who -together- set a new standard for parenting. Words are simply not enough to describe the role you both played towards my life and career success. Thank you for all the motivation, inspiration and empowerment you have given me.

I also dedicate this work to my grandmother, who always remembers me in her prayers and continuously draws the smile on my face, and my brothers, my source of enthusiasm and happiness.

ACKNOWLEDGMENTS

I would like to take this opportunity to express my gratitude for my thesis adviser, Dr. Sami Zhioua, who not only guided me throughout the idea generation, experiments' execution and write-up of my thesis, but also for being the main reason why I decided to pursue my master's degree and career in Information and Cyber Security. My thanks extend to the committee members, Dr. Moataz Ahmed and Dr. Sajjad Mahmood for their continuous and valuable support. I do appreciate the KFUPM community for the educational experience that empowered me as a person, playing a pivotal role in my career and future.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

NAME:	Ahmed Abouollo
TITLE OF STUDY:	Canvas Fingerprinting: A State of the Art
MAJOR FIELD:	Security and Information Assurance
DATE OF DEGREE:	February, 2019

Web applications' operators have many motivations to track users and gather as much user identifiable information about them as possible. With about three quarters of the web pages including third party trackers, privacy continues to be one of the major concerns to web users. This research is mainly focused on a tracking technique called Canvas fingerprinting. Canvas is an HTML element that allows to dynamically render 2D shapes and bitmap images. It is one of several technologies introduced in HTML5 making it a serious alternative to Flash which is being discontinued because of its multitude of security vulnerabilities. Interestingly, Canvas can be used for fingerprinting browsers, and hence for tracking users. This thesis is a state of the art of Canvas fingerprinting in which we explore the functionality the Canvas element was originally introduced for, provide updated results about the prevalence of Canvas in the web, its distinguishing capabilities, its positive (at-

tack detection) and negative (attacks on privacy) use-cases. The major findings of this study is that Canvas is very common among web applications (1 out of 4 websites are using Canvas for all constructive and destructive purposes), while it was recently reported that the Canvas usage for fingerprinting is (10.44%), which demonstrates the significance of Canvas and cost of disabling it, and by optimizing Canvas elements we could improve the distinguishing entropy from 0.49 to 0.83, which exceeds the distinguishing capability of the 18 non-Canvas fingerprints we studied. We performed several assessments that show the reliability of this reported entropy including cross data validation and benchmarking with other major fingerprinting datasets in the literature. The two novel constructive use cases we propose for utilizing Canvas are Using Canvas fingerprinting for the detection of fake accounts creation on web applications, and for session hijacking prevention. We studied the effectiveness of both techniques through an empirical study and the implementation of a proof of concept, respectively. We finally explored five scenarios where Canvas fingerprinting can be exploited to attack users privacy even when using different devices or visiting separate web applications.

ملخص الرسالة

الاسم: أحمد محمد عبد اللطيف أبوعلو

عنوان الرسالة: مراقبة مستخدمي الويب باستخدام تقنية الكانفاس

التخصص: أمن المعلومات

تاريخ الدرجة العلمية: نوفمبر ٢٠١٨

العاملون على تطبيقات الويب لديهم الكثير من المحفزات لتتبع مستخدمي تلك التطبيقات والحصول على أكبر قدر من معلوماتهم. وبما أن حوالي ثلاثة أرباع صفحات الويب تحتوي على أطراف ثالثة للتعقب، أصبحت الخصوصية من أولويات مستخدمي الويب. هذا البحث يركز على إحدى طرق التتبع وهي بصمة الكانفاس (Canvas). تقنية الكانفاس والتي أضيفت حديثا بصورة رسمية في لغة ترميز النصوص التشعبية (HTML) تسمح بعرض الأشكال ثنائية الأبعاد والصور النقطية ديناميكيًا. تعد تقنية الكانفاس واحدة من التقنيات التي تمثل بديلا بارزا لتقنية الفلاش والتي سيتوقف دعمها بحلول عام ٢٠٢٠ لكثرة الثغرات الأمنية فيها. من المثير للاهتمام أنه تم اكتشاف طريقة يمكن بها استخدام تقنية الكانفاس لأخذ بصمات من المتصفحات تمكن أصحاب المواقع من تتبع مستخدمي الإنترنت.

هذه الرسالة تدرس تقنية الكانفاس من جوانب عدة: منها دراسة مدى انتشار التنقنية في الويب، وقدرتها على تمييز المستخدمين بهدف تتبعهم، إضافة إلى طرق وجدوى استخدام هذه التقنية لأغراض بناءة (كالكشف عن الهجمات الإلكترونية) وأغراض سلبية (كتهديد خصوصية مستخدمي الإنترنت).

من أهم استنتاجات هذه الدراسة إثبات أن واحدا من كل أربعة مواقع على الإنترنت تستخدم تقنية الكانفاس بشكل عام لجميع الأغراض السلبية منها والبناءة (بينما نسبة استخدام هذه التقنية لأغراض التتبع هي ١٠,٤٤٪ كما تم ذكره سابقا)، وأنه بتحسين عينات الكانفاس المستخدمة لأخذ بصمات متصفحات الإنترنت استطعنا رفع معامل العشوائية (entropy) - والذي يستخدم لقياس القدرة في تمييز البصمات الإلكترونية - من ٠,٤٩ إلى ٠,٨٣.

CHAPTER 1

INTRODUCTION

Privacy on the web continues to be a concern for many web users, as the majority of web services strive to track users and get hold of as much of their personal identifiable information as possible [7]. This information can include the products web users purchase, the people with whom they interact, the websites they frequently visit, and their areas of interest. Although the purpose of gathering such information is commercial in many of the cases, it can easily jeopardize users' privacy. For example, a web application may be able to tell that the owners of two different accounts belong to roommates if the two accounts are being accessed regularly from the same computer [8]. According to a study conducted on 850,000 web users and 144 million web page visits, 77.4% of the web page loads contain implanted trackers [9]. The same study pointed out that Google and Facebook trackers are the most commonly used ones, as Google Analytics was found in 46% of the included websites and Facebook Connect in 21.9% of them. If a user visits multiple web applications where a third party tracker is planted, the tracker can create a detailed user profile and keep updating it with every visit. With this, trackers can obtain highly personal information, like health information, financial situation, religious and political views. It has been reported that third-party trackers monitor to know the users who visit a web page that belongs to Mayo Clinic, which provides information about HIV tests, and by this the trackers can tell if a user clicks the button to arrange an appointment [10].

A quantitative survey conducted by the market research agency CG Selecties on 924 respondents concluded that there is a relationship between consumers' privacy and their behaviour [11]. People are generally concerned about their privacy, and it is critical to them to be able to stay in control of their personal data and the information they decide to share. The more privacy concerns the consumers have, the more negative their attitude is towards the collection of their data. For example, the consumers would adjust the settings of their smart phones and disable the location services, sacrificing the features that comes with it, in order to prevent their browsing from being tracked. In other words, when the consumers do not feel in control of their data, it can trigger a negative behavior and the consumers will resist and try to avoid it.

1.1 Tracking Motivations

There are many motivations why a web application would track users and gather identifiable information about them. These motivations go beyond the targeted advertisements to serve the web users giving them personalized and seamless experience. This includes showing relevant content when a web user searches for general search key words in a search engine, as well as giving the right movie suggestions in Netflix, products recommendations in Amazon, and business reviews in Yelp [12]. However, such personalized experience facilitates price discrimination. Hannak et al. found evidence that nine out of sixteen e-commerce websites included in their study present customizing prices as a result of user tracking [12]. It has been shown that the web users' geographical location can change the returning price by up to 166%, the wealth by up to 400%, and the referring website by up to 50% [13]. Mikians et al. performed a crowd sourcing study where 340 users surfed the web, while information was gathered using a browser extension, showing that many retailers give product prices for returning users that range between 10% and 30% [14]. Moreover, tracking users and gathering identifiable information helps in assessing the financial credibility of people. Some financial organizations use as much as 8,000 data points in order to evaluate a loan application, including data from the social network accounts of a person, e-commerce websites, and user behaviors like whether a user spent adequate time reading about the loan details before applying [15]. Further more, Insurance companies like Allfinanz and TCP LifeSystems can utilize user information to determine the probability of a person getting a disease or making an accident based on information coming from credit card spending, magazine subscriptions and customer surveys [16]. Also, web tracking serves government surveillance. It has been reported that government spying agencies like National Security Agency (NSA) utilize the third party trackers to spy on individuals, and even know several aspects including the user location data [17]. These agencies can also have the authority to force companies to share web users records [17].

Previous work in the literature revealed that even when web clients clear cookies or use private browsing, they are still not completely protected from being tracked. In fact, clearing cookies for every request may be an identifying behavior that distinguishes the hosts from others. Web users who wish not to be tracked can take more tracking counter measures such as modifying the default settings, using proxies, using anonymous routing, or a combination of these techniques [18].

1.2 Evolution of Web Tracking

Web tracking techniques are increasingly sophisticated and are made more pervasive, intrusive, and persistent. The authors of [19] point out that using IP addresses to track web users is inadequate for several reasons. One of the reasons is that if more than one user are browsing the web behind the same Network Address Translation (NAT) domain, they are very likely to have the same IP address, which makes it difficult to distinguish between them. Another reason is that when a user uses Tor, an adversary will see different IP addresses for different requests from the same user. Cookies have been used for tracking purposes for a long time. However, because cookies lack persistence, and because users can erase them easily, adversaries designed more persistent techniques including Evercookies and the use of cookie syncing in conjunction with Evercookies [4]. Flash cookies (also known as Evercookies) can be utilized to regenerate HTTP cookies removed by the user via a technique called cookie re-spawning. Interestingly, a study found that 41 out of the 100 most popular websites stored flash cookies with content matching with the regular cookies [20]. The re-spawning technique starts when a user visits the website in which the technique is implemented, which in turn creates an ID and stored it in both the HTTP cookies and the flash cookies. When the user erases the HTTP cookies, the website reads the value stored in the flash cookie and places a new HTTP cookie with the same ID.

1.3 Fingerprinting and Canvas

Fingerprinting works by requesting a browser's version and configuration information that are available to the web application upon request, and connecting this information with a device or user account to identify users when they visit later. In his research about device fingerprinting, Eckersley investigated the extent to which web browsers are vulnerable to fingerprinting [21]. The researcher implemented a fingerprinting algorithm that retrieves browsers version and configuration information, and according to the collected sample of fingerprints, he showed that if we pick a fingerprint randomly, not more than 1 out of 286,777 of other browsers would share the same fingerprint. The same research suggests that fingerprints can be used as global identifiers. They can be thought of as cookies that cannot be deleted except with a large enough configuration change to break the fingerprint. The research also suggests using a combination of a fingerprint and IP address to regenerate cookies in a similar way to the website that use flash cookies to regenerate HTTP cookies [4], or use the IP address and the fingerprint on their own to replace the functionality of cookies.

It has been reported that an authentication technology has been widely used for e-commerce and online bank to minimize fraud, which come at the price of privacy. When a bank user logs in to the bank website, they provide their username and password, but in addition to that, the website checks some fingerprints to identify the device and make sure the person owning the account is the same one accessing, assuming that most users use the same device for banking [8]. Fingerprinting in these technologies works by requesting a computer for some details such as browser type, language, time zone, cookie ID, flash ID, and IP address. If there is a sufficient number of matches, the account is granted access.

A later research published in 2016 proposed a fingerprinting script composed of 17 attributes. This research was based on a data set of 118,934 fingerprint collected by the website AmIUnique.org [1]. The aim of launching the website was to gather as many samples of fingerprints as possible to study the diversity of fingerprints. The research showed the effectiveness of each attribute in distinguishing browsers by reporting the number of distinct values and unique values of each attribute in the dataset. The research also demonstrated the effectiveness of the fingerprinting of mobile devices in spite of the less number of fonts and plugins available. In addition, several research papers reported that Canvas was one of the most distinguishing attributes of the fingerprint [4] [1] [3].

1.4 Contributions

The contributions of our work can be summarized as follows:

1. We studied Canvas presence in the top 100,000 Alexa sites to find out that almost one out of four websites use Canvas in the landing page of the website. This indicates that blocking the Canvas feature for the sake of privacy on the browser level would result in functionality failure on a significant number of websites, and hence researchers need to find other creative solutions to protect users privacy.

2. We designed 23 Canvas samples with various components and characteristics and analyzed the impact of each on the distinguishing capability of Canvas fingerprinting, and came up with observations and guidelines for designing optimal fingerprinting Canvas samples.

3. We proposed an enhanced Canvas sample to use in fingerprinting users, which increased the normalized entropy for Canvas fingerprinting with the widely used sample from 0.490 when calculating the entropy in our dataset (and 0.491 in the AmIUnique dataset with the same sample [1]) to a normalized entropy of 0.837 when using our enhanced sample. This normalized entropy achieved by our enhanced Canvas sample is higher than any Canvas and non-Canvas fingerprint used in AmIUnique [1] and Panopticlick [21].

4. We proposed a technique that utilizes Canvas fingerprinting for the detection of fake accounts creation on web applications. We carried out an empirical

study that showed the effectiveness of the proposed technique, resulting in 6.67% of false positives and 7.44% false negatives.

5. We proposed a technique that utilizes Canvas fingerprinting for the prevention of session hijacking. We built a fully functioning proof of concept to demonstrate the effectiveness of the technique.

6. We studied five practical scenarios and explained how Canvas fingerprinting could be taken advantage of in order to attack users' privacy even when using different devices or visiting separate web applications.

This thesis is structured as follows: Chapter 2 reviews the literature related to Canvas prevalence, distinguishing capability and major fingerprinting datasets. Chapter 3 studies the Canvas HTML element, its history, support, what a web developer can use Canvas to build, and the extent to which Canvas is present in web applications, in order to understand the impact of disabling the feature for the sake of privacy. Chapter 4 presents an empirical study to demonstrate the possibility of enhancing the Canvas fingerprinting samples to exceed the distinguishing capabilities of other fingerprinting techniques. Chapter 5 proposes and assesses novel constructive applications to utilize Canvas fingerprinting: the detection of fake accounts creation on web applications, and the prevention of session hijacking. Chapter 6 spots the light on five different scenarios where Canvas fingerprinting can be exploited to attack the privacy of web users, and discusses some issues Tor browser users face when visiting websites with Canvas content. Chapter 7 points out how we validated our findings, what threat we see to the validity of our results, demonstrates how to replicate our empirical studies, and finally concludes the research and proposes future work.

CHAPTER 2

LITERATURE REVIEW

This chapter sheds the light over the relevant literature to our research. It starts by explaining web fingerprinting as a tracking technique, how it is classified, and pointing out the main focus of our research. It then explains why we chose to study Canvas as opposed to the other fingerprinting techniques. Afterwards, it reviews the relevant work done on Canvas prevalence and the earlier findings, and how these results complement our findings. This chapter also discusses the enhancement of Canvas samples used for fingerprinting over time, and explains the gap left unanswered, which we address in future chapters. Finally, this chapter reviews three large scale fingerprinting datasets, pointing out the targets, findings and drawbacks of each, in order to have benchmarks that we can compare our findings to.

2.1 Web Fingerprinting

When studying web fingerprinting for tracking users, it is important to note that fingerprinting is not a single tracking technique. In fact, web fingerprinting can be done using a combination of fingerprinting techniques. Those techniques can be of the same type or different types that are combined to achieve better tracking of web users. Hence, it is important to understand the various types of fingerprinting techniques, and how they are classified in the literature. In their research, Upathilake et al. studied web browser fingerprinting and focused on providing a logical way of classifying the different techniques [22]. The outcome of their classification was the following four categories. First, Browser Specific fingerprinting which is associated to the browsing environment. Algorithms belonging to this category utilize java or flash to obtain browser specific information that is used as fingerprints like resolution, User Agent, list of fonts, HTTP Accept, and list of plugins. The weaknesses mentioned are the instability of these fingerprints as small changes like installing a new font or changing the monitors resolution can affect the fingerprint [23], and the inability of distinguishing identically configured devices [24]. The Second category is Canvas fingerprinting which utilizes pixel data of rendered images at the web client device, which is the main focus of our research. The third category is JavaScript Engine fingerprinting that performs conformance testing such as Sputnik test suite [25] and matches the failed tests of a browser to the browser version known for failing these test, as suggested by Mulazzani et al. [26]. This fingerprinting technique can detect modified user-Agent strings and even identifies Tor browser users [27]. Finally, Cross-browser fingerprinting is similar in concept to the browser specific fingerprinting except that it uses JavaScript to obtain the information, instead of relying on Java or Flash. Hence, it shares the same weakness of the inability to distinguish devices with the same configurations. Finally, Cross-browser fingerprinting is similar in concept to the browser specific fingerprinting except that it uses JavaScript to obtain the information, instead of relying on Java or Flash. Hence, it shares the same weakness of the inability to distinguish devices with the same configurations. It is important not to confuse this last type (i.e. cross browser fingerprinting) with the cross-device tracking studied extensively by Brookman et al., which uses a combination of web tracking techniques including several fingerprinting and non-fingerprinting techniques aiming to track users and link users on their different devices [28]. Companies specialized in cross device tracking work on developing graphs which link users to their different devices to provide these graph to other parties who subscribe to this service and are interested in tracking users.

The previous fingerprinting techniques of different classifications are applicable to both desktops and mobile devices. However, since mobile devices are more likely to have the exact same hardware and software configurations, and due to the lack of plug-ins in mobile devices, it became necessary find alternatives to these fingerprinting techniques. Bojinov et al. utilized the components more relevant to mobile devices to come up two implementations to track web users: one through analyzing the mobile device accelerometer behavior and calibration errors which do not require specific user permissions to obtain, while the other by emitting sounds using different frequencies and recording them back to analyze values like sound amplitude and distortions to use as fingerprints [29]. Canvas fingerprinting is the major focus of our research, as this research analyze it from various aspects like prevalence, distinguishing capability, constructive and destructive use cases. In our research, we also considered many other fingerprints that are part of browser fingerprinting and cross-browser fingerprinting in the previous classification in order to confirm the findings of other research and to compare their distinguishing capability with Canvas. We do not consider JavaScript Engine fingerprinting, accelerometer fingerprinting nor spearker-microphone fingerprinting.

2.2 Fingerprinting via Canvas

In this research, our focus is to study the different aspects related to one type of fingerprints from the previous classifications (i.e. Canvas fingerprint) from the perspective of prevalence, distinguishing capability, positive and negative use cases. There are many reasons why we select to study Canvas fingerprinting as a tracking technique. Some of which are the several positive characteristics reported by Mowery [3], including consistency in the value of the fingerprint coming from the same device and browser, the high entropy, transparency to the web users, and not requiring permission to obtain. Another reason why we chose to study Canvas fingerprinting is that despite the observed drop in the entropy of several fingerprinting characteristics due to the changes in the web trends, like the entropy drop for the list of plugins with the decreasing support especially in mobile devices [1] [5], we continue to see stable and high distinguishing capability for Canvas fingerprint. Moreover, Alaca et al. studied 29 types of fingerprints and classified according to their properties, to show that fingerprinting by Canvas has high repeatability of fingerprint values, consumes low device resources and is ranked medium in terms of the distinguishing information it carries [30], despite being vulnerable to client spoofing like any fingerprint that requires client side execution of JavaScript which returns the output to the server.

In addition to that, we find Canvas to be one of the tracking techniques that has high potential to enhance, as we can see in the different research in reviewed in Section 2.2.2 where more complex Canvas samples are used over time for better entropy, and as we extensively show in Chapter 4 using 23 different samples to study how the components of Canvas can affect the entropy. The following two sections review the prevalence of Canvas as reported in the literature by different research between the years 2014 and 2016, and show the enhancement done to the Canvas samples that are used in fingerprinting over time.

2.2.1 Canvas Prevalence

After Mowery discovered Canvas as an effective, high entropy fingerprinting technique to track users without any permission required in 2012 [3], websites started to deploy Canvas fingerprinting gradually. That is why several researchers started looking at Canvas prevalence among websites on the web. In an effort to find out the prevalence and effectiveness of post cookies tracking techniques, Acar et al. studied the extent to which the HTML Canvas feature is used in fingerprinting and tracking users [4]. In their experiment, they used an instrumented Firefox browser that was modified to log function calls that could indicate a website is fingerprinting users through Canvas, and crawled the top 100,000 Alexa sites looking for function calls which are used in fingerprinting. The results indicate that over 5.5% of sites ran scripts to fingerprint users through Canvas on their home pages. Acar filtered out false positives by removing websites that do not have both Canvas drawing and pixel retrieving calls coming from the same URL as they are not likely to be used for tracking, the Canvas images with very few pixels as they are not effective enough to be used for tracking, and the Canvas with images in a lossy compression format that can render differently at different times.

In 2017, Englehardt et al. presented on of the most comprehensive studies of web tracking ever done [31]. In their study, they covered as many as 15 types of tracking measurement, including stateful and stateless tracking methods. Fingerprints studied included Canvas Fingerprinting, Audio Context fingerprinting and Battery API Fingerprinting. They utilized the an open source tool that they built for web privacy measurement and named OpenWPM **(https://github.com/citp/OpenWPM)**. It is noteworthy that Englehardt et al. built on the 2014 study [4], and added more elimination criteria to reduce false positives. This explains some of the reduction in the Canvas fingerprinting prevalence reported Englehardt et al. Some of the added criteria to consider the

Canvas a false positive is that the text written into Canvas has less than 10 distinct characters since the Canvases used in fingerprinting are likely to have more characters for better distinguishing capability, and that the script uses an API call like addEventListener, save or restore that are typically used for legitimate purposes. The reported prevalence of Canvas was 4.03% for the top 10,000, and 2.61% for the top 100,000 Alexa websites, respectively.

In 2016, Le et al. performed an empirical study to detect the prevalence of Canvas fingerprinting among the top 10,000 Alexa websites [32]. Their research was a pioneering one in analyzing and reporting the websites that use obfuscated tracking along with those that do not. Obfuscated tracking refers to a commonly used process to hide the JavaScript code making it very difficult to read, modify and reuse, as well as less visible to pattern detecting techniques [33]. There are even free web services that can be used to perform obfuscation and mimic the JavaScript code producing the same exact execution outcome [34] [35]. The methodology Le et al. proposed and utilized is based on a dynamic analysis of the JavaScript API calls performed by the browser in comparison with the source code sent to the browser by the website in order to detect obfuscated tracking. When applying the proposed methodology, it was found that 10.44% of the top 10,000 Alexa websites use Canvas fingerprinting in the home page of the website, out of which 2.25% use obfuscation techniques.

In our research, we use the results of Le et al. in [32] to calculate the estimated prevalence of Canvas used for legitimate purposes, as they are the most recent and close in time to our experiments, which means that the Canvas deployment would have changed the least compared to other earlier research. Also, the results of [32] proved to be more accurate as in the comparison done by Le et al. showing that in the top 100 websites, the methodology used by Englehardt et al. in [31] results in false positives (6 websites that contain Canvas were missed out of 8). Table 2.1 summarizes the finding of previous research on the Canvas fingerprinting prevalence over time among the top Alexa websites.

Table 2.1: Canvas Fingerprinting Prevalence According to Different Research

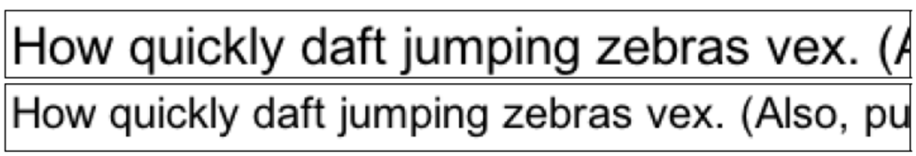
Author	Year Published	Websites Crawling Date	Prevalence (top 10K Websites)	Prevalence (top 100K Websites)
Acar et al. [4]	2014	May 2014	4.93%	5.73%
Englehardt et al. [31]	2017	January 2016	4.03%	2.61%
Le et al. [32]	2017	April 2016	10.44%	-

Chapter 3.2 of this research complements the previously reported results about Canvas fingerprinting prevalence to show the overall prevalence of Canvas on the web for all purposes, and provides an estimation of how much of this prevalence is in fact for legitimate purposes.

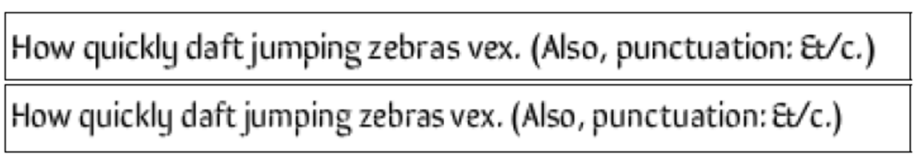
2.2.2 Canvas Distinguishing Capability

Reviewing the research done on Canvas fingerprinting over time, we notice that the Canvas samples being used for fingerprinting have been increasing in complexity. When discovering Canvas as a fingerprinting technique in 2012, Mowery gathered Canvas fingerprints from 300 web users using the 5 samples in Figure 2.1 [3].

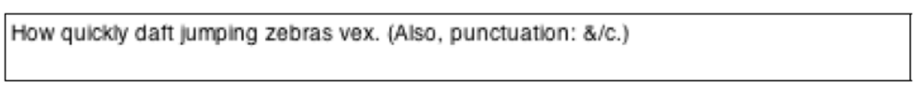
Mowery used a pangram (i.e. a sentence that contains all the alphabetical letters) with some punctuation in the 5 samples. The font in the first two samples was Arial with font sizes of 18pt and 20pt. The font in the two following samples was Sirin Stencil (imported from Google Web Fonts) with font sizes of 12pt and 15pt. Finally, the last sample's font is a a fake font name that was used to study the effect of the fallback handling of fonts by different browsers. These Canvas samples are relatively simple and result in low entropy with distinct fingerprint values between 43 and 50 from the 300 users who were fingerprinted. Meaning that the fingerprints gathered are able to distribute the users into 43 to 50 groups of users (depending on which sample is used), each group has a distinct fingerprint value.



`text_arial` (top) and `text_arial_px`



`text_webfont` (top) and `text_webfont_px`



`text_nonsense`

Figure 2.1: The 5 Canvas Samples used by Mowery [3]

In 2014, Acar et al. performed an empirical study to show the percentage of the top 100,000 Alexa websites which use Canvas for the sake of Fingerprinting (refer to Chapter 3.2 for more details) [4]. They found out the Canvas sample used the most among the websites is the one in Figure 2.2. This sample contains perfect pangram text drawn twice using two colors and a fake font name (to include the text fallback factor). The sample also included the Unicode smiling face character, and an orange rectangle. The research does not calculate the entropy of the sample as their main target is to calculate the Canvas prevalence for fingerprinting purposes.

Figure 2.2: The Canvas Sample reported by Acar [4]

The fingerprinting dataset gathered by Laperdrix et al. in 2016 via the website AmIUnique used the same Canvas sample to fingerprint users, except that the two texts do not overlap as in Figure 2.3 [1]. The normalized Shannon entropy of this Canvas sample was 0.491 when calculated on the dataset of over 118,000 fingerprints (more analysis on this dataset in Section 2.3). This sample became more commonly used than the samples used by Mowery (Figure 2.2) as their complexity produces more entropy, which increases the distinguishing capability.

Figure 2.3: The Canvas Sample used by Laperdrix in AmIUnique [1]

The most recent large scale fingerprinting study performed in 2018 by Gomez-Boix et al. using one of the top French websites was able to gather fingerprints from over two million user visits [5]. Gomez-Boix et al. used the Canvas sample in Figure 2.4, which is the most complex Canvas sample we have seen in a large scale fingerprinting research. Although the sample is more complex than the one in Figure 2.3, it produced an entropy of 0.407, which is less than the entropy produced by the less complex sample. This can be explained by several reasons: the differences between the percentages of Operating Systems and browsers included in each of the studies (which is what we refer to as the market share bias in Table 2.2), or the changes in web trends over time affecting the entropy of a fingerprinting techniques. Here arises the importance of performing the distinguishing capability analysis on the same dataset of web users using many Canvas samples, to understand the impact of introducing more complexity while fixing all the other variables, as we do in Chapter 4.

Figure 2.4: The Canvas Sample used by Gomez-Boix [5]

2.3 Significant Fingerprinting Datasets

Several research papers studied the effectiveness of fingerprinting, and the extent to which it can be used to uniquely identify web users. For benchmarking, we chose three large scale datasets to compare our findings to. These datasets span over 8 years since the discovery of fingerprinting, to also have an idea of any trends occur in the distinguishing capability of the various fingerprints. In 2010, Eckersley performed the first large scale study of device fingerprinting, using the website Panopticlick [36]. The number of fingerprinting samples gathered by the website were initially 1,043,426, and reduced to 470,161 fingerprints by several stages of preprocessing and filtering, to remove some records affected by a client side bug, and to eliminate any bias [21]. In his research, Eckersley was able to show the large diversity of devices over the web, and that by collecting device and browser specific information such as HTTP headers, list of plug-ins and attributes gathered by JavaScript, it was possible to uniquely identify over 94% of the browsers at the time of performing the research according to his gathered samples. It is noteworthy that the data gathered by this website is biased, as the visitors are mostly technical people who are privacy aware.

In 2016, Laperdrix et al. performed another large scale experiment to study the effectiveness of web fingerprinting, how the user uniqueness has changed from 2010, and how the fingerprinting techniques have developed [1]. This was the first extensive experiment that studies fingerprinting on mobile devices in addition to desktops. They built the website AmIUnique [37], using which they gathered fingerprints from over 118,000 user visits, each of them were fingerprinted using 17 attributes. Unlike Eckersley (2010) [21], Laperdrix et al. included Canvas as a fingerprinting attribute in their research, since Canvas fingerprint was only discovered by Mowery et al. in 2012 [3]. Laperdrix et al. reported that 89.4% of the clients visiting the AmIUnique website are unique. They analyzed the differences in the distinguishing capability of the various fingerprinting attributes over time. Some major findings are the significant and continuous reduction in the browser plugins, and showing that Canvas is one of the most discriminating attributes. They performed a simulation that estimates the impact of the removal of plugins and the usage of generic HTTP headers to reduce the uniqueness of desktop fingerprints by 36%. Table 4.8 compares the normalized entropy for some of attributes that vary significantly between desktop and mobile devices, according to their dataset. One of the downsides of this research is the bias towards the privacy aware community, who tend to visit the website more than the average users of the web. Table 2.2 presents the bias in the AmIUnique dataset and in our dataset (as analyzed later in Chapter 4.3) in terms of the proportion of operating systems in the web against the actual, as reported by StatCounter in October 2018 [38]. It is noteworthy that the bias in our dataset is less than the dataset obtained from the Website AmIUnique, as all our percentages are the closer to the actual OS market share.

Gomez-Boix et al. (2018) targeted to overcome this bias of privacy aware website visitors by collecting and analyzing over 2 million fingerprints gathered from

Table 2.2: Bias in the Various Datasets in the proportion of Operating Systems Against the Actual

OS Type	AmIUnique [1]	French Website [5]	Our Dataset	Actual [38]
Windows	57%	82%	56%	36%
Linux	15%	<1%	1%	<1%
Mac	13%	5%	8%	6%
Android	5%	9%	19%	40%
iOS	4%	2%	15%	13%

one of the top 15 French site to target broader audience [5]. A noteworthy finding is that the percentage of unique fingerprints has reduced to 33.6% from 89.4% in 2016, showing a significant reduction in the ability to distinguish web users in about 2 year time span. Gomez-Boix et al. explained that this drop in the distinguishing capability was due to the less biased dataset towards privacy aware people, as well as the reduction of plugins in desktops, showing that the changes happening to web technologies nowadays improve user privacy. Gomez-Boix et al. shows the the fingerprinting attribute that is the most capable of distinguishing mobile devices is Canvas, while it is the list of plugins in desktops. They also showed experimentally that the non-unique fingerprints in desktops are significantly more fragile than mobile devices, meaning that if a browser fingerprint is not unique, it is more probable to become unique by changing a fingerprinting attribute in desktops than in mobile devices. Table 2.2 also lists the operating system proportions in the dataset of [5], and how it compares to our dataset and the actual operating system market share, presenting less bias than the AmIUnique dataset [1]. Finally, Table 2.3 summarizes the Shannon normalized entropy reported of the various fingerprinting attributes as reported by the tree benchmarking datasets, which our research refers to in the upcoming chapters.

Table 2.3: Shannon Normalized Entropy of the Three Major Fingerprinting Datasets in Literature

Fingerprinting Attribute	Panopticlick (2010) [21]	AmIUnique (2016) [1]	Hiding In The Crowd (2018) [5]
Platform	-	0.137	0.057
Do Not Track	-	0.056	0.091
Timezone	0.161	0.198	0.008
List of plugins	0.817	0.656	0.452
Use of local/session storage	-	0.024	0.002
Use of an ad blocker	-	0.059	0.002
WebGL Vendor	-	0.127	0.109
WebGL Renderer	-	0.202	0.264
Available fonts	0.738	0.497	0.329
Canvas	-	0.491	0.407
Header Accept	-	0.082	0.035
Content encoding	-	0.091	0.018
Content language	-	0.351	0.129
User-agent	0.531	0.580	0.341
Screen resolution	0.256	0.290	0.231
List of HTTP headers	-	0.249	0.085
Cookies enabled	0.019	0.015	0.000

CHAPTER 3

HTML CANVAS: OVERVIEW AND PREVALENCE

This chapter consists of two major sections. The first section gives an overview about Canvas starting from its history and support. It describes the various functionality Canvas was introduced for, showing why it is a serious candidate to replace Flash with real life examples. It also demonstrates how JavaScript is used to manipulate Canvas and shows the corresponding output. Finally, it shows what API methods are being utilized for the tracking purposes, which is the main basis for the following chapters. The second section of this chapter includes an empirical study to understand the extent to which Canvas is prevalent in the web for both the legitimate and tracking purposes, in order to understand the cost of disabling Canvas to protect users' privacy. This is to complement the literature which only focuses on the Canvas prevalence for tracking. The second section starts by defining the approach and experimental setup, reporting initial results, revising the approach and reporting the final findings.

3.1 Canvas Feature Background

Canvas was introduced with HTML5, and it is one of the most powerful features of it. Canvas works as a placeholder on which shapes, images and text can be drawn on the fly. There is Canvas specific API that can be used for many purposes including drawing shapes like lines, curves, circles, rectangles and fill them with colors, gradients or patterns. Drawing and manipulating text, changing the font properties and position, manipulating images, processing videos, making smooth animations and developing games are all possible using the Canvas powerful API [6]. Geary has used Canvas to implement several applications that can be otherwise implemented using Flash. For example, the image panning application in Fig. 3.1, a paint application that runs on browsers and iPads, as well as animations and games [6]. We observed several scientific web applications using Canvas to build animated solar system model, for example, to teach kids, while other websites implemented user interactive 3D application that shows how chemical atoms are bound to construct molecules. Countless websites built games with complex physics, while others included animated analog clocks built by Canvas. With the expected death of Flash in 2020 [39], Canvas seems to be a strong possible replacement for many of its capabilities including drawing, animations, and interactivity. We see popular websites shifting from Flash to Canvas such as **http://www.speedtest.net** which utilizes Canvas in its new Flash-free version and moved the original Flash version of the website to **http://legacy.speedtest.net/**. Fig. 3.2 shows how the Canvas version looks in the Flash-free SpeedTest. Canvas is supported in most of the major browsers nowadays. The support has started from version 4.0 of Google Chrome, version 9.0 of Internet Explorer, version 2.0 of Mozilla Firefox, version 3.1 of Safari and version 9.0 of Opera [40].

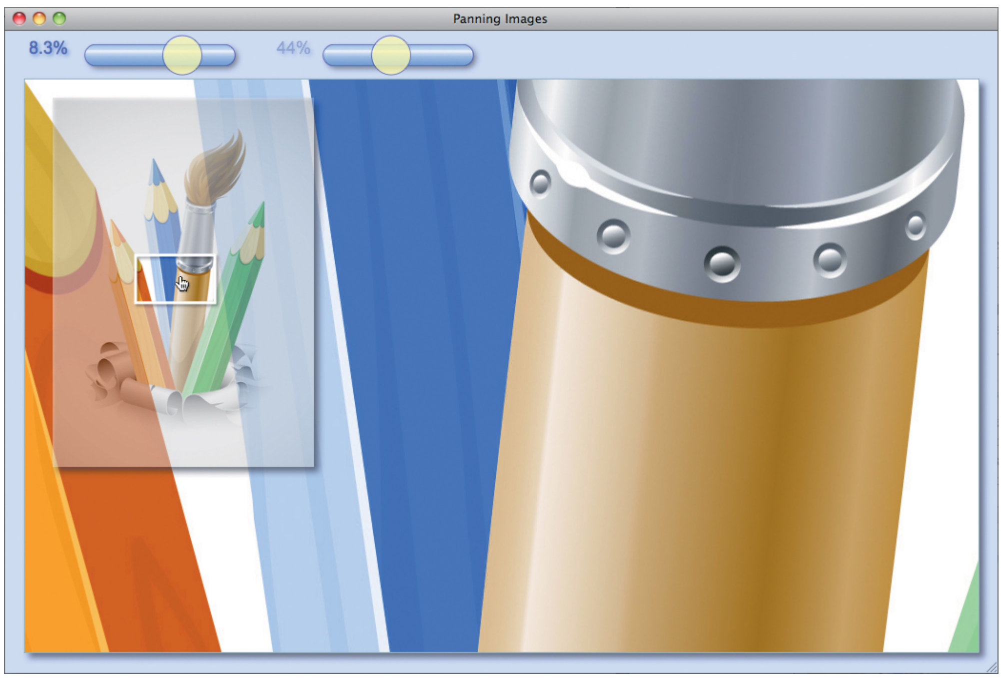


Figure 3.1: Image Panning Implemented Using Canvas API [6]

3.1.1 Concrete Canvas Example

Canvas is an HTML container for a Context, which is used to draw graphics. This is done by retrieving the HTML tag <`canvas`> using the JavaScript method `document.getElementById()`, then getting the Canvas context by the method **getContext('2d')**, and using the context to draw content to the Canvas. After

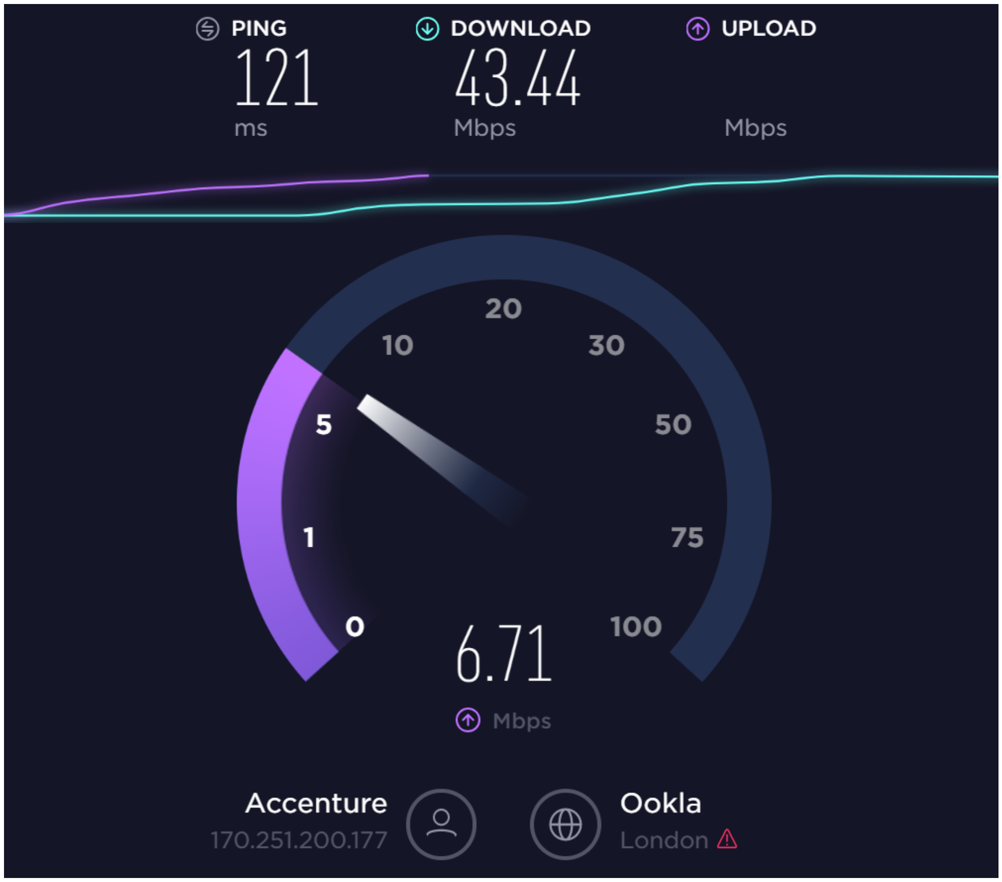


Figure 3.2: Flash-free Version of **speedtest.net**

getting the context of the Canvas, JavaScript is used to set the values of the attributes such as the font color attribute of the drawn text (**fillStyle**) and the size and type attribute of the font (**font**), and so on [6]. Listing 3.1 shows an example on how to get a Canvas element, retrieve the context, change the attributes and finally draw shapes and text. The output of the example is displayed as in Fig. 3.3.

The attributes of the Canvas context such as the text font, fill style and shadow can be updated dynamically upon the user's interaction with the website. In addition to that, the attributes can be saved and restored any time by JavaScript methods if there is a need to temporarily change the attribute values. Canvas

Listing 3.1: Example of an HTML Canvas Element Retrieved and Manipulated Using JavaScript

```
<canvas id='canvas' width='500' height='330' style='border:1px solid;'>
</canvas>

<script type='text/javascript'>

var canvas = document.getElementById('canvas'); // Creating the Canvas Variable
var context = canvas.getContext('2d'); // Retrieving the Canvas Context

context.font = '38pt Times New Roman'; // Changing the font type and size
context.fillStyle = 'brown'; // Changing the fill color to brown
context.strokeStyle = 'Green'; // Changing the stroke color to brown

context.fillText('Canvas Example', 60,60); // Filling Text
context.strokeText('Canvas Example', 60,60); // Stroking Text

context.lineJoin = 'round'; // Making the stroked rectangle with rounded edges
context.lineWidth = 30; //Changing the width of the stroked rectangle
context.strokeStyle = 'cornflowerblue'; // Choosing stroke color
context.strokeRect(85 /*x*/, 100 /*y*/, 140 /*width*/, 180 /*height*/); //Stroking a rectangle

context.fillStyle = 'burlywood'; // Choosing fill color
context.fillRect(250, 140, 160, 100); //Filling a rectangle

</script>
```

applications can be built to detect user interactions such as mouse and keyboard events and respond accordingly. For example, event listeners can be added for actions like **onmousedown**, **onmouseup**, **onmouseout** and **onmousemove** to

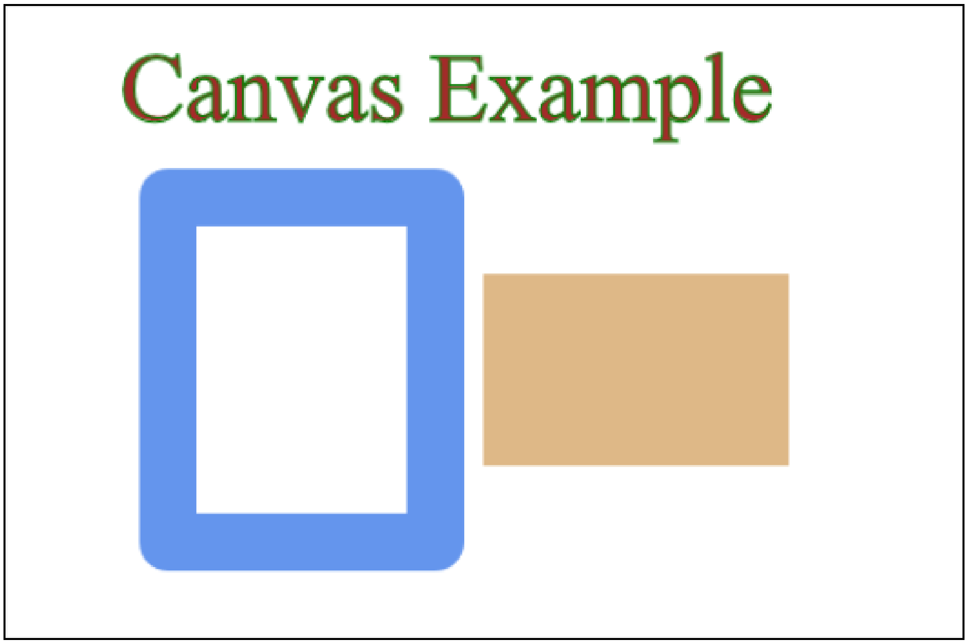


Figure 3.3: Output of The Canvas Manipulation

handle mouse events, as well as **keydown**, **keypress** and **keyup** for keyboard events. Canvas has a coordinate system with the origin at the upper left corner. The X-axis increases to the right and the Y-axis increases to the bottom.

3.1.2 Attack Surface

Canvas is a bitmap, not an image HTML element. So, if the web developer would like to give access to users to retrieve or download the Canvas as an image, a method named **toDataURL()** can be used to get a Data URL and assign it to an image HTML element in the **src** attribute. This enables the user of an online paint application built by Canvas, for example, to download a snapshot of the drawing they made. Another method named **toBlob()** allows to store the Canvas in a file. These two JavaScript methods make it possible to return the

Canvas data as it was rendered on the client machines' web browsers. However, these methods are found to be implemented and used by many web applications to substitute the common techniques of tracking users such as cookies [4] [1] [3] [23]. The applications proposed and implemented in this research also use one of these methods (i.e. **toDataURL()**) to retrieve the Canvas data and use it as a fingerprint for detecting attacks like session hijacking and fake accounts creation on web applications, as well as in attacking users' privacy.

3.2 Canvas Prevalence

In order to establish a common understanding of how important and widely used Canvas is, we defined a different target and approach to those research experiments previously published and pointed out in Chapter 2.2.1. Our target is to find out the number and percentage of websites using Canvas, both for legitimate purposes which the Canvas feature was initially meant for, or for defeating users' privacy. Either of these ways of using Canvas is convincing for us to continue our research in Canvas, given that the more Canvas is used legitimately, the more difficult it is for users to block the Canvas feature on the browser level, as this will result on functionality failure of a significant number of websites. Also, the more Canvas is being used to track users, the more researchers need to find solutions to protect users' privacy. So we decided to look for the percentage of top Alexa websites that use Canvas for all purposes, both for legitimate and user tracking purposes, and rely on the results published in [32] to estimate the percentage of which that are using Canvas for legitimate purposes.

3.2.1 Experimental Setup

In our initial definition, we considered a website to be using Canvas if we find the HTML element <canvas> in the main HTML source code of the website. To know the Canvas prevalence, we developed a script that scrapes the top 100,000 Alexa websites looking for the ones that contain the <canvas> in the HTML. The script is written in Python and uses Selenium web driver **(http://www.seleniumhq.org/)**. Selenium is a tool that runs and drives a web browser application, like Mozilla Firefox in our case, through a programming interface, and is designed for testing web applications. Choosing Selenium was to mimic real browser's behavior, and to overcome anti scrapping techniques that are implemented in some websites where other libraries fail to circumvent, as well as Selenium helped in avoiding decoding errors we faced when trying other libraries. Once a website is visited by the web browser, the source HTML is passed to Beautiful Soup. Beautiful Soup is a Python Library that provides methods to enable parsing, searching and navigation through a document to extract the needed information from it **(https://www.crummy.com/software/BeautifulSoup/)**. Our script used Beautiful Soup to look for the <canvas> HTML element, and if found, it marks the website as a website that uses Canvas. Algorithm 1 is explains the logic of our initial scrapping script.

We ran our Scrapping script between 27-December-2017 and 3-January-2018

Algorithm 1 Initial Scrapping Algorithm for Canvas Usage

Input: list of n websites to be scrapped on the current server (i.e. subset of the top 100,000 Alexa sites)

Output: Percentage of websites containing <canvas> in HTML source

Each server concurrently executes the following code for its corresponding list of websites:
Read a CSV file and construct a list of n websites to be run on the current server
$x \leftarrow 0$ {i.e. number of websites containing Canvas}
$y \leftarrow 0$ {i.e. number of websites failed in reading}
$z \leftarrow 0$ {i.e. number of websites successfully read}
for $i \leftarrow 0, i < n, i \leftarrow i + 1$ **do**
    Initialize a selenium webdriver instance
    Set the *page_load_timeout* value to 60 seconds
    Pass the URL to the browser through the webdriver
    Get HTML source from the browser
    if website loading time passes *page_load_timeout* **then**
        $y \leftarrow y + 1$
        Continue to the next for loop iteration
    else
        $z \leftarrow z + 1$
    end if
    Pass HTML source to BeautifulSoup parser
    Search for "<canvas>" in HTML
    if "<canvas>" is found **then**
        $x \leftarrow x + 1$
    end if
end for
return $x, y, z, x/z$ {i.e. percentage of websites containing <canvas> in HTML source}

on 10 Google Cloud servers in parallel, each server visiting 10,000 websites from the list of the top 100,000 Alexa sites. Each of the servers ran Windows Server 2016 as the operating system, on a dual core Intel Xeon CPU with a speed of 2.30 GHz and a RAM size between 7.5 and 10 Gigabytes. The Internet connectivity is stable on the Google Cloud servers, with more than 270 Mbps of download speed and over 85 Mbps of upload speed. The reasons why we decided to run the script on Google Cloud servers are the easiness in selecting the exact hardware specifications that we need and pay for them only during the execution of the script, the Internet connectivity stability, the ability to execute the scripts in parallel to reduce the execution duration, and to reduce the number of blocked sites by the local ISP, since the cloud servers used were in the United States, where less sites may be blocked.

3.2.2 Initial Results

The results of running the scrapping script showed that 9.34% of the top 500 websites had the <canvas> element in their main website page. This percentage kept decreasing to be 5.37% for the top 1000 websites, 2.83% for the top 10,000 and 2.45% for the top 100,000 websites. Table 3.2 shows the number of websites attempted to be visited by the script, the number of websites successfully read, the number of websites failed in reading, the number of websites using the <canvas> element, and the percentage of websites using the <canvas> element from the successfully read websites. It is noteworthy that around 10% of the website were not successfully read because of reachability problems or due to the 60 seconds explicit timeout that we set for reading a website. Table 3.1 points out the percentage of websites containing Canvas elements among the top 500, 1000, 10,000 and 100,000 websites.

Table 3.1: Usage of Canvas Elements by top 500, 1K, 10K and 100K Websites

Top 500	9.34%
Top 1000	5.37%
Top 10K	2.83%
Top 100K	2.45%

Table 3.2: Portion of the top 100K Websites that Contain <canvas> Element in the HTML by 10K intervals

Alexa Sites' Range	[1-10K]	(10K-20K]	(20K-30K]	(30K-40K]	(40K-50K]	(50K-60K]	(60K-70K]	(70K-80K]	(80K-90K]	(90K-100K]	Total
Successfully Read	9,182	9,030	9,020	8,901	8,915	8,950	8,933	8,989	8,950	9,017	89,887
Reading Failed	818	970	980	1,099	1,085	1,050	1,067	1,011	1,050	983	10,113
Contains Canvas	260	226	220	193	218	215	232	210	211	220	2,205
Canvas Prevalence %	2.83%	2.50%	2.44%	2.17%	2.45%	2.40%	2.60%	2.34%	2.36%	2.44%	2.45%

Table 3.3: Portion of the top 100K Websites that Use Canvas by 10K Intervals According to the Revised Definition

Alexa Sites' Range	[1-10K]	(10K-20K]	(20K-30K]	(30K-40K]	(40K-50K]	(50K-60K]	(60K-70K]	(70K-80K]	(80K-90K]	(90K-100K]	Total
Successfully Read	9,182	9,088	9,018	8,922	8,894	8,932	8,971	8,995	8,993	9,002	89,997
Reading Failed	818	912	982	1,078	1,106	1,068	1,029	1,005	1,007	998	10,003
Contains Canvas	2,016	2,116	2,030	2,029	2,151	2,174	2,320	2,238	2,266	2,297	21,637
Canvas Prevalence %	21.96%	23.28%	22.51%	22.74%	24.18%	24.34%	25.86%	24.88%	25.20%	25.52%	24.04%

3.2.3 Revised Definition and Findings

After investigations, we realized that many websites that use Canvas do not necessarily contain the <canvas> element in the main HTML source, but rather in the iframes loaded with the website. Some websites even do not even contain a <canvas> element, as they use the JavaScript method **createElement("canvas")** to create the Canvas in a <script> element or in an imported JavaScript file. Thus, we revised the initial definition and correspondingly the script to consider a website to be using Canvas if:

- A Canvas element is found in the source HTML or any associated iframe.
- The method **createElement("canvas")** or **createElement('canvas')** is found in any <script> element the source HTML or any associated iframe.
- The method **createElement("canvas")** or **createElement('canvas')** is found in any imported JavaScript file whether from the main source HTML or any associated iframe.

We realize that the script detection can be circumvented by passing a string variable instead of "canvas" in the argument of the method, but this is a limitation that can be accepted. Algorithm 2 shows how the logic of our revised scrapping script looks like.

We ran our revised scrapping script between 12-January-2018 and 16-January-2018 on the same 10 Google Cloud servers. The results showed that an average of 24.04% of the top 100,000 websites use Canvas. Table 3.3 shows the number of websites attempted to be visited by the script, the number of websites successfully read, the number of websites failed in reading, the number of websites using Canvas according to the revised definition, and their percentage from the successfully read websites. Again, around 10% of the website were not successfully read because of reachability problems or due to the 60 seconds explicit timeout that we set for reading a website. Table 3.4 points out the percentage of websites containing Canvas according to the revised definition among the top 500, 1000, 10,000 and 100,000 websites. Table 3.5 presents our findings in conjunction with [32] to get an estimation of the Canvas prevalence by type of usage, showing that over 50% of the websites we crawled and reported to be using Canvas are using it for legitimate purposes. Since this calculation is a direct subtraction, it considers a website that uses Canvas for both legitimate and tracking purposes only in the tracking side, which is a safe assumption that may only reduce the prevalence for legitimate usage. Besides being the first reported Canvas prevalence statistic for legitimate purposes that is reported in the literature, the percentages reported in Table 3.5 help quantify the potential high cost of completely blocking Canvas. This may occur if users decide to use browser plugins to block Canvas from appearing in the loaded web pages, or if web browsers decide to disable Canvas for the sake of protecting user's privacy. In either cases, web users will face partial or complete functionality failures when visiting 24% of all the web applications, out of which over 50% are solely using Canvas for legitimate purposes, according to our estimation. Therefore, users will not continue to have seamless experi-

Table 3.4: Usage of Canvas According to the Revised Definition by top 500, 1K, 10K and 100K Websites

Top 500	20.80%
Top 1000	20.04%
Top 10K	21.96%
Top 100K	24.04%

ence when visiting a huge amount of websites (1 out of 4), leading to the need of rebuilding new versions of the same websites that are Canvas-free.

Table 3.5: Estimated Canvas Prevalence by Type of Usage

Author	Year Published	Websites Crawling Date	Prevalence (top 10K Websites)
Our Research (Overall Canvas Usage)	2018	January 2018	21.96%
Le et al. [32] (for tracking purposes)	2017	April 2016	10.44%
Our Research (for legitimate purposes)	2018	-	11.52%

It is noteworthy that several websites do not fingerprint the user on the landing page, so they were not detected or included in our calculations by definition. An example of these websites is https://amiunique.org/ which was initially built by Laperdrix et al. to carry out their research on fingerprinting techniques [1]. Another observation is that many websites like Google and its variants use Canvas in some days but not in others. Since our reported results were based on the empirical study we carried out in conjunction with another research findings [32], it is important take into consideration the following dependencies: there is a duration of 21 months separating the execution of our crawling script and the crawling done in [32], which may cause some impact in the web trends that affect our reported percentage of Canvas prevalence for legitimate purposes. Moreover, calculating the Canvas prevalence for legitimate purposes was through a direct subtraction of the overall usage minus tracking usage, which introduced a limitation when a website uses Canvas for both legitimate and tracking purposes, as it has been only considered in the tracking side.

3.2.4 Algorithms Explanation

Algorithm 1 aims to visit each of the top 100,000 Alexa websites searching for the HTML element <Canvas> in the source code, and and outputs the percentage of websites that include the element in the main HTML page. It takes as an input a CSV file with the list of websites. The algorithm starts by distributing the list of websites to be visited among the servers available, and each participating server reads the corresponding list from the CSV file. Afterwards, the script starts three counter on each server and initializes the counter to 0. The counters are x (the number of websites containing <Canvas> in the HTML source code), y (the number of websites failed in reading, so they do not contribute positively or negatively in the percentage of Canvas prevalence) and z (the number of websites successfully read by the server). The percentage of Canvas prevalence will be calculated later by dividing x/z.

The main body of Algorithm 1 (i.e. lines 6-22) consists of a loop and two if statements, where each loop is for a single website visit. every iteration of the loop initializes a Selenium webdriver instance and passes a website's URL to it in order to mimics a user visit to the website. The instance is set to time out if the website does not load within 60 seconds, increments the y counter (websites failed in reading), and continues to the next loop iteration. Otherwise if the website is read successfully the z counter increments (successfully read websites' counter) and the website HTML source is passed to the BeautifulSoup parser, which in turn searches for the HTML element <canvas>. If found, the x counter increments. Once the loop passes over all the websites a server is assigned to visit, the algorithm returns the values of the three counters x, y and z, as well as the percentage of websites containing <canvas> (i.e. x/z).

Algorithm 2 shares the same input, initial steps and counters as Algorithm 1. The differences are in the Output and the main body of the loop. The output of Algorithm 2 is the percentage of websites using Canvas in the HTML or JavaScript, in all the frames loaded with the home web page of the website. For every iteration of the main loop body (i.e. lines 6-35), a Selenium webdriver instance is initialized with a website's URL. If the website times out, the y counter in cements (failed websites), and continues to the next loop iteration. Otherwise if the website is read successfully the z counter increments (successfully read) and a list of the loaded frames is constructed. Another internal loop iterates over the list of loaded frames with the website, retrieving all the JavaScript loaded (whether in loaded files or inside <script> tags). If any of the HTML pages in any of the frames include a Canvas tag, or any of the JavaScript components use the method to create a Canvas element, the x counter increments. Once the outermost loop passes over all the websites a server is meant to visit, the algorithm returns the values x, y, z, and x/z.

3.3 Conclusion

In this chapter, we shed the light over the HTML5 Canvas element, the feature used for developing interactive content on web applications. The wide support of Canvas among browsers, the examples given of applications developed using Canvas, and the websites switching their content to newer Canvas versions from what previously was developed in Flash are all facts that make it possible to observe the trend of the increasing adoption of Canvas. This chapter also gives a concrete example of how to retrieve a Canvas element using JavaScript, manipulate it by drawing content, then shows the actual output of how it looks after rendering on the web page. Afterwards, it was pointed out which API function can be used to fingerprint users, enabling the constructive and destructive applications we propose and validate in Chapters 5 and 6.

The literature studied the prevalence of Canvas fingerprinting, and reported the significant increase overtime since the discovery of Canvas as a fingerprinting mechanism in 2012 [3]. A recent study published in 2017 reported the over 10% of the top 10,000 Alexa websites use Canvas for fingerprinting. To complement the reported findings, we studied the Canvas prevalence for all purposes, to help realize the cost of disabling Canvas entirely if someone wishes to protect themselves from being tracked. To do this, we developed and ran the scrapping algorithms explained in this chapter on 10 rented servers which visited 100,000 websites to find out that about 1 out of 4 websites use Canvas, out of which 11.52% use Canvas for legitimate purposes.

Algorithm 2 Revised Scrapping Algorithm for Canvas Usage

Input: list of n websites to be scrapped on the current server (i.e. subset of the top 100,000 Alexa sites)

Output: Percentage of websites using Canvas in HTML or JavaScript, whether in the main frame or any iframe

Each server concurrently executes the following code for its corresponding list of websites:
Read a CSV file and construct a list of n websites to be run on the current server
$x \leftarrow 0$ {i.e. number of websites containing Canvas}
$y \leftarrow 0$ {i.e. number of websites failed in reading}
$z \leftarrow 0$ {i.e. number of websites successfully read}
for $i \leftarrow 0, i < n, i \leftarrow i + 1$ **do**
    Initialize a selenium webdriver instance
    Set the *page_load_timeout* value to 60 seconds
    Pass the URL to the browser through the webdriver
    Construct a list F of all iframes loaded including the default frame
    if website loading time passes *page_load_timeout* **then**
        $y \leftarrow y + 1$
        Continue to the next for loop iteration
    else
        $z \leftarrow z + 1$
    end if
    for each $f \in F$ **do**
        Retrieve HTML source loaded in frame f
        Retrieve and construct a list S of all JavaScript tags' and files' content loaded in frame f
        Pass HTML source and list S of frame f to BeautifulSoup parser
        Search for "<canvas>" in HTML
        if "<canvas>" is found **then**
            $x \leftarrow x + 1$
            Continue to the next outermost for loop iteration
        else
            for each $s \in S$ **do**
                Search for "createElement("canvas")" in s
                if "createElement("canvas")" is found **then**
                    $x \leftarrow x + 1$
                    Continue to the next outermost for loop iteration
                end if
            end for
        end if
    end for
end for
return $x, y, z, x/z$ {i.e. percentage of websites using Canvas according to the revised definition}

CHAPTER 4

CANVAS DISTINGUISHING CAPABILITY ANALYSIS

In Section 2.2.2 of the Literature Review, we observed a trend in the Canvas samples used for fingerprinting, which is the increased complexity over time. Observing this trend helps realizing that Canvas is one of the tracking techniques that has high potential to enhance. Moreover, there are many pros of Canvas like the consistency in the fingerprint value coming from the same device and browser, the high entropy, the transparency to web users, and not requiring permission to obtain [3]. All of these reasons are the basis of dedicating this chapter to study the distinguishing capability of different Canvas samples. To achieve that, we designed and carried out an empirical study with the following targets: drafting guidelines on how to enhance the Canvas samples used to fingerprint users, providing strong Canvas samples in terms of distinguishing users, and comparing the Canvas samples to other widely used fingerprinting techniques.

4.1 Experimental Setup

To achieve our targets, we built a website and hosted it publicly. The website asks the visitors to fill out a form with the specifications of the device and browser from which they are currently browsing. The website asks the user to select the category of the device she is using, then shows different forms and instructions for the different types of devices. For example, the iPhone and iPad specific form does not ask the user to provide the Graphics Card Type and Version or the RAM size because it is less intuitive to the user to obtain such pieces of information, and we can obtain these details by the device model information the user provides. Also, the user instructions on how to obtain the specifications information are platform specific. On each of the form fields, we provided a link that opens the steps on a new browser tab. Fig. 4.1 shows the landing page of the website.

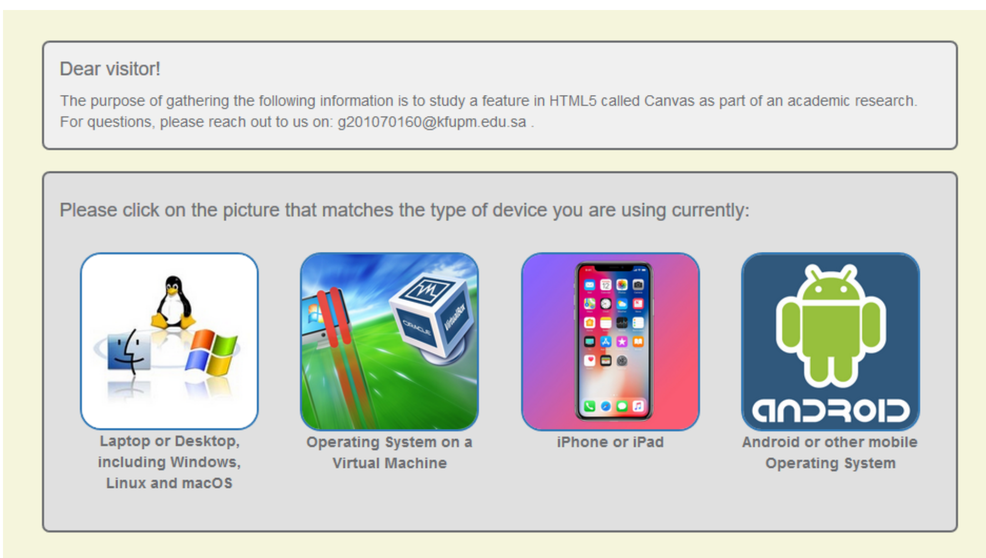


Figure 4.1: Landing Page of the Data Gathering Website

The specifications we asked for included the browsers type and version, operating systems type and version, device model, whether the browser used is installed in a virtual environment, Graphics Card Type and Version, Graphics Card Memory, RAM Size, CPU Type and number of cores. The step-by-step instructions provided to the visitors explained how to gather these details from the operating systems: Windows, Linux, MacOS, iOS and Android, and from the browsers: Google Chrome, Mozilla Firefox, Microsoft Edge, Internet Explorer and Opera.

Once the user submits the devices specifications, the website retrieves many other Canvas and non-Canvas fingerprints along with the user provided details. This allows for correlating the device specifications to the fingerprints retrieved. The non-Canvas fingerprints gathered are listed in Table 4.1 with corresponding values gathered from a Chrome browser on a laptop running Windows 10 as examples of what the fingerprints could look like. The reason why we gathered the non-Canvas fingerprints is to see how our dataset compares to the datasets other research on fingerprinting previously used, such as Laperdrix et al. [1] and Eckersley [21] in terms of entropy, especially for the fingerprints with high entropy like the list of fonts, plugins, and user agent. In addition, gathering non-Canvas fingerprints allows us to compare our top Canvas fingerprints to other that are being used currently in the wild, and to set a benchmark that we try to reach by our enhanced Canvas fingerprints. These fingerprints became handy during the data filtering and helped us being confident that a certain person has submitted the online form more than once. The fingerprinting code that gathered the fingerprints in Table 4.1 was originally developed by Valentin Vasilyev and updated by Dave Alger, who has published the source code with a free license to use **(https://codepen.io/run-time/pen/XJNXWV)**.

We also designed 23 Canvas samples and gathered them to fingerprint users. While designing the samples, we took into account making them diverse. We did this by adding various components to the different samples like text, emojis, shapes, backgrounds, pictures. Even for those samples with similar components, we changed other characteristics such as the number of characters and emojis, types of fonts, colors, adding shadows. For several Canvas samples, we also combined several components in a single Canvas. Table 4.2 summarizes the characters of each of the 23 Canvas samples we gathered, and Appendix A includes the actual rendered samples along with statistics on how many unique samples, distinct samples, and how many characters there are in the Data URL version of each sample.

Table 4.1: The Samples of the Non-Canvas Fingerprints We Gathered

Fingerprint	Description	Example	Shannon Entropy	Normalized Entropy
Browser	Represents the browser name and major version	Chrome 64	2.743	0.304
Flash	A string that includes the Flash Player version	N/A	0	0
Canvas	A string in Base 64 representing how pixels are rendered on a device. The printed image in this case: 	data:image/png;base64,iVBORw0KGgo AAAANSUhEUgAAASwAAACWCAYA AABkW7XSAAAgAElEQVR4Xu	4.426	0.490
Connection	Specifies if the connection is wifi, celullar or undefined	Undefined	1.442	0.160
Cookie	A boolean that indicates whether cookies are enabled	True	0.020	0.002
Display	Actual & available width & height, and color depth	24—1366—768—1366—728	5.271	0.584
Font Smoothing	A boolean that indicates if font smoothing is enabled. Smoothing aims to avoid pixelation across resolutions	True	0.839	0.100
Fonts	Includes the list of fonts detected at the client browser	Agency FB—Arial Black—Bodoni MT —Calibri Light—Castellar—Colonna MT —Consolas—Constantia—Copperplate Gothic Light	5.809	0.643
Form Fields	Returns all form fields a user can input data into, to detect if certain clients try to inject extra form fields into the website	url=https://s.codepen.io/boomerang/ iFrameKey-20452a66-c145-f241-f16d-fb053c67cd18/index.html	1.497	0.166
Java	A boolean that indicates whether Java is enabled	False	0.309	0.034
Language	The current browser, user and system language settings	lang=en-US—syslang=—userlang=	1.101	0.122
Silverlight	A string that includes the Silverlight version	N/A	0.135	0.0150
Operating System	The operating system on which the browser is running	Windows NT 4.0—32 bits	3.020	0.334
Time Zone	A number that represents the relative time of a client	3	2.761	0.306
Touch	A boolean that indicates whether touch is enabled	False	0.802	0.0888
True Browser	Looks specifically for the real application name	Safari	0.648	0.072
Plugins	Includes the list of plugins detected at the client browser	chrome pdf plugin—chrome pdf viewer—native client—widevine content decryption module	2.769	0.307
User Agent	An HTTP header string that is passed from a client to the web server to tailor responses according to the capabilities of a client	mozilla/5.0 (windows nt 10.0; win64; x64) applewebkit/537.36 (khtml, like gecko) chrome/64.0.3282.140 safari/537.36—Win32—en-US	6.181	0.684
Comprehensive	Hash of the combination of all attributes in this table	2876715415	8.490	0.940

Samples and Characteristics	Canvas 1	Canvas 2	Canvas 3	Canvas 4	Canvas 5	Canvas 6	Canvas 7	Canvas 8	Canvas 9	Canvas 10	Canvas 11	Canvas 12	Canvas 13	Canvas 14	Canvas 15	Canvas 16	Canvas 17	Canvas 18	Canvas 19	Canvas 20	Canvas 21	Canvas 22	Canvas 23
Has text	X									X	X	X	X	X	X	X	X		X	X	X	X	
Has Emoji	X																	X	X	X			
Has all alphabetical letters										X	X	X	X	X	X	X	X		X	X	X	X	
Has more than 20 emojis																		X	X	X			
Several fonts																			X	X			
Has Shapes		X	X		X		X	X	X										X	X	X	X	
Has several shapes		X	X		X		X	X	X										X	X			
Has Shadows									X										X	X			
Text/Shapes in diff. colors			X		X		X	X	X										X	X	X	X	
Has Gradient				X	X	X	X	X	X										X	X			
Contains picture																							X
Background entirely Covered				X	X				X										X	X			
Canvas String Length	5,304	17,164	15,076	74,000	63,592	52,940	44,240	28,064	96,072	29,528	32,108	16,876	26,000	30,376	35,644	36,616	31,716	29,784	226,416	228,708	11,196	29,080	228,708
Distinct Values	119	36	60	89	105	103	126	117	132	96	99	83	88	94	84	83	92	152	317	276	79	135	275
Unique Values	62	13	22	32	43	36	52	50	63	42	47	42	43	47	39	38	48	89	232	186	41	76	185
Shannon Entropy	5.570	3.141	4.677	5.510	5.767	5.788	6.078	5.922	6.078	4.999	5.002	4.225	4.730	4.846	4.661	4.653	4.375	5.934	7.884	7.558	4.438	5.857	7.551
Normalized Entropy	0.617	0.348	0.518	0.610	0.638	0.641	0.673	0.656	0.673	0.553	0.554	0.468	0.524	0.536	0.516	0.515	0.484	0.657	0.873	0.837	0.491	0.648	0.836

Table 4.2: Characteristics of the Canvas Samples Gathered

4.2 Dataset

The website containing the online form was distributed publicly to our acquaintances, the academic community, and social networks. In addition, we posted a task to fill out the online form on Mechanical Turk [41]. Mechanical Turk is a marketplace for human intelligence where on-demand workforce is available to complete tasks that cannot otherwise be automated. The task we posted forwards the people who accept the task to our website to fill out the online form with the details of the device and browser from which they are browsing. Once they fill out the form, our website gives them a unique random number to enter in the Mechanical Turk website as a confirmation that they completed the task, and they get paid accordingly. People were asked to only provide one submission from the same device, so that the number of distinct values is not impacted by similar fingerprints submitted from the same device.

We reviewed and analyzed the data submitted by users on our website to remove duplicates and incorrect input. Several factors helped us in determining whether certain submissions are duplicated. The indications we looked at are that the user provided details are exactly same, the user writing style and submission timeframe are very close, all or most Canvas and non-Canvas fingerprints are the same, the submission is done from the same IP address. We eliminated all the submissions where the input was determined to be duplicated submissions from the same devices. Afterwards, we reviewed the submissions looking for inaccurate data input. A third review of the data was done to ensure the entered data is consistent and spelled correctly in order to be able to filter the data by specific fields. Some submissions included data that appears to be accurate, but not all the fields were filled. We accepted some of these submissions and included them in the entropy calculations, but did not use them in any specific analysis that required the missing information. After the data filtering took place, the number of submission that were considered in the study has decreased from 701 to 524 submissions.

4.3 Analysis

Tables 4.3 and 4.4 show the number of users' submissions per operating system and browser respectively. several of the submissions were missing those details, and hence, the number of samples does not sum up to 524. It is interesting to note that the devices running the different versions of Windows constructed 56% of the dataset, and the submissions done from the Chrome browser were 70% of the dataset.

The bar chart in Fig. 4.2 shows the total number of user submissions in the leftmost bar, which is the maximum number of distinct values a fingerprint can achieve. The higher number of distinct values a Canvas sample (or any fingerprint) has, the better it is in fingerprinting and in distinguishing different devices. Fig. 4.2 also shows the ability of each Canvas sample in Appendix A to distinguish users into different groups via fingerprinting. The two rightmost bars in the bar chart represent the number of distinct values retrieved by the Canvas sample commonly

Table 4.3: Number and Percentage of Submissions per Operating System

Operating System	Number of Samples	Percentage
Windows 10	145	28.9%
Windows 7	99	19.8%
Android (several versions)	95	19.0%
iOS (several versions)	76	15.2%
MacOS (several versions)	41	8.2%
Windows 8	31	6.2%
Linux (several distributions)	6	1.2%
Windows XP	4	0.8%
Windows Vista	3	0.6%
ChromeOS	1	0.2%

Table 4.4: Number and Percentage of Submissions per Browser

Browser Type	Number of Samples	Percentage
Chrome	359	69.7%
Safari	60	11.7%
Firefox	59	11.5%
Internet Explorer	13	2.5%
Opera	10	1.9%
Edge	6	1.2%
UC browser	5	1.0%
Dolphin	3	0.6%

used in the wild, which is also used as part of the multi-attribute fingerprinting algorithm in Table 4.1, as well as the number of the distinct values retrieved by the comprehensive multi-attribute fingerprinting algorithm.

Canvas samples 19, 20 and 23 seem to be the best three samples in distinguishing devices. Although 19 is able to distribute devices into the highest number of groups (i.e. 317) based on the Canvas fingerprint, we do not recommend using it because this sample is not very persistent. When we loaded the website on the same browser using the same device, we got more than one fingerprint value at different times. When investigated further, we found out that Google fonts are not always loading correctly although the same setting was in place. Canvases 20 and 23 are among the most powerful in distinguishing devices. Our preference is Canvas 20, since it does not require loading a picture from the Internet like Canvas 23 does, and therefore requires less downloading time.

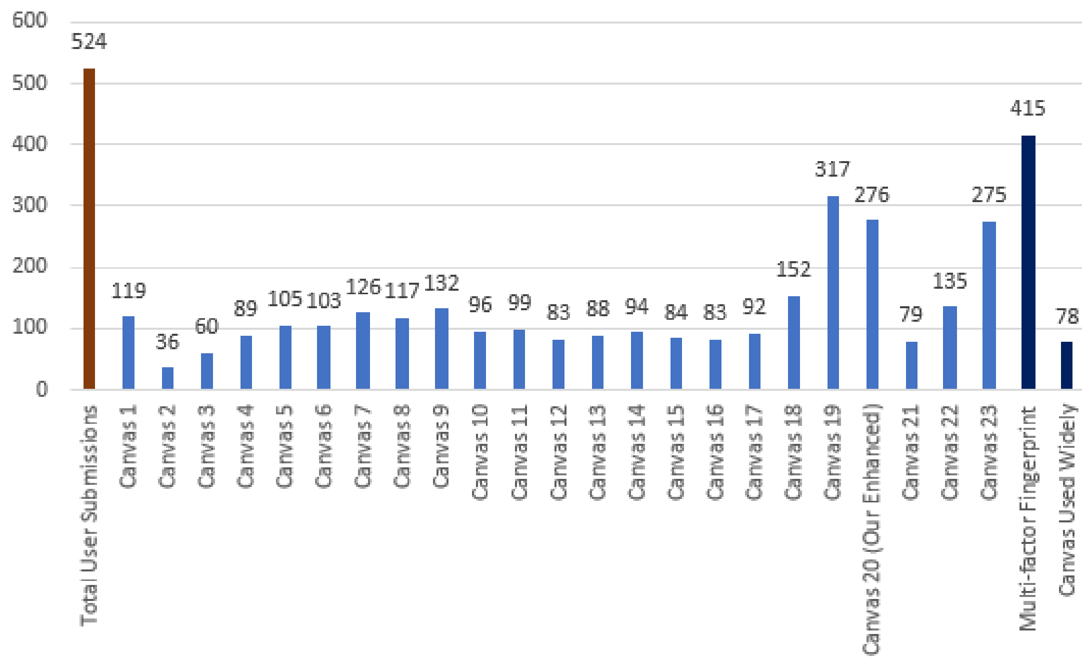


Figure 4.2: Number of Distinct Canvas Values per Sample

As in the major fingerprinting research papers, we use Shannon entropy to indicate the amount of information a fingerprint gives. The fingerprint is better and more identifying when the entropy is higher. We calculated the Shannon's entropy using the formula:

$$Entropy = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

Where $P(x_i)$ is the probability of the value x_i occurring according to our dataset, and n is the total number of distinct fingerprint values in the dataset. The probability is calculated for a fingerprint value x_i by dividing the number of times the same value of the fingerprint appeared by the total number of values (i.e. valid user submissions) in the dataset.

In order to eliminate the factor of the dataset size, and to compare our obtained fingerprints with other datasets like AmIUnique and Panopticlick, we calculated the Normalized Shannon's Entropy by dividing the entropy obtained using the previous equation by the maximum possible entropy $log_2(N)$ where all devices' fingerprints are unique (N represents the entire size of the dataset). The Normalized Shannon Entropy results always in a value between 0 and 1. Practically on large datasets, it is very unlikely to see any fingerprinting technique with Normalized Shannon Entropy reaching 1, which indicates that the technique is ideal in fingerprinting web users. However, reaching the maximum normalized entropy is theoretically possible if and only if all the values appearing in the dataset being analyzed are unique. Even if a dataset includes distinct values that appear exactly the same number of times each, this condition does not apply and the Shannon entropy would be less than 1. For illustration, let D_1 and D_2 be 2 datasets of size 8 where $D_1 = \{v_1, v_2, v_3, ..., v_8\}$, and $v_1 \neq v_2 \neq v_3 \neq ... \neq v_8$. While $D_2 = \{v_1, v_1, v_2, v_2, v_3, v_3, v_4, v_4\}$, and $v_1 \neq v_2 \neq v_3 \neq v_4$. We demonstrate below how D_1 (which contain values that are all unique) result in Normalized Shannon Entropy that equals 1 and how D_2 (which contain distinct values repeated the same number of times) result in Normalized Shannon Entropy that is less than 1:

$$\text{Shannon entropy of } D_1 = -\sum_{i=1}^{8} P(v_i)\log_2 P(v_i) = -8[P(v_i)\log_2 P(v_i)]$$

$$= -8[(0.125)(-3)] = 3$$

Since the maximum possible entropy of a dataset with 8 values is $log_2(8) = 3$, then the Normalized Shannon Entropy of dataset $D_1 = 1$.

$$\text{Similarly, Shannon entropy of } D_2 = -\sum_{i=1}^{4} P(v_i)\log_2 P(v_i)$$

$$= -4[P(v_i)\log_2 P(v_i)] = -4[(0.25)(-2)] = 2$$

Since the dataset has the same number of values, the maximum possible entropy is $log_2(8) = 3$, then the Normalized Shannon Entropy of dataset $D_2 = 0.667$. The advantage of using the Normalized Shannon Entropy is that it is dependent on the distribution of probabilities, and does not take into consideration the dataset size, making it fair to compare distinguishing capability results of a fingerprinting technique across datasets of different sizes. In addition, Normalized Shannon Entropy is widespread among the other web fingerprinting research and other well known datasets in the literature. With this, we find the Normalized Shannon Entropy to be the most fit measurement to use. Table 4.5 compares the normalized entropy of seven of the most common fingerprints including Canvas on separate datasets: Panopticlick [21], AmIUnique [1], Hiding In The Crowd [5] and our research. It also shows how our enhanced Canvas sample compares to the commonly used Canvas sample (as well as other non-Canvas fingerprints) in terms of entropy.

Table 4.5: Normalized Entropy of the Major Fingerprinting Attributes on Three Datasets

Attribute	Our Research (2018)	Hiding In The Crowd (2018)	AmIUnique (2016)	Panopticlick (2010)
User agent	0.684	0.431	0.570	0.531
List of plugins	0.307	0.452	0.578	0.817
List of fonts	0.643	0.329	0.446	0.738
Display	0.584	0.341	0.277	0.256
Time Zone	0.306	0.008	0.201	0.161
Cookies enabled	0.002	0.000	0.042	0.019
Canvas	0.490	0.407 (Different Sample)	0.491	-
Enhanced Canvas (Sample no. 20)	0.837	-	-	-

The most significant entropy differences between our dataset and other previous research is observed in the Display and List of plugins attributes. AmIUnique explained the drop of 0.24 in the entropy of the List of Plugins comparison to Panopticlick to be a result of the absence of plugins in the mobile devices, especially with the increasing use of mobile devices to surf the web, as well as because of the stopped support for the old NPAPI plugin architecture since 2015 on Google Chrome [1]. The entropy of the list of plugins continues to drop by 0.27 compared to AmIUnique. AmIUnique's dataset includes 13,105 mobile devices fingerprints out of 118,934, constituting a percentage of 11% of the overall dataset, while 34.2% of the fingerprinted devices in our dataset are mobile devices, which can explain the drop in our List of Plugins entropy. The observed increase of 0.31

in the entropy of the display attribute in comparison to AmIUnique can be explained by the difference in the information gathered by the different algorithms. While AmIUnique gathers the screen's width, height and color depth, we gather the available width and available height in addition.

It is interesting to note that the Canvas fingerprint gives almost identical entropy in both AmIUnique dataset and our dataset. Panopticlick's research occurred before Mowery et al. discovered Canvas fingerprinting in 2012, and hence it did not include Canvas in their study. Our enhanced Canvas sample (number 20) showed a significant increase in the entropy and the distinguishing capability to successfully exceed all other fingerprints studied in this research, in AmIUnique and in Panopticlick. These results give us high confidence that Canvas samples can be enhanced to be more effective in distinguishing devices than the other existing fingerprints.

4.4 Results and Validation

To understand the extent to which the reported entropy of our enhanced Canvas sample is reliable, and the ability of our dataset to represent the population of web user, we did the following:

- Compare the entropy of a Canvas sample that is common between our research and another large dataset.
- Perform Cross Data Validation

As table 4.5 points out, Our research and the AMIUnique dataset have a Canvas sample in common. When calculating the Shannon Entropy of this Canvas sample according to our dataset, it was 0.490, while the calculated entropy for the same Canvas sample is 0.491 according to AmIUnique [1]. This gives confidence that our dataset is valid and representative of similar large datasets, and can be used to calculate the entropy of other Canvas fingerprinting samples to determine their distinguishing capability.

To perform the Cross data validation, we randomized the 524 fingerprints gathered from every Canvas sample, and each visit was fingerprinted with multiple Canvas and non-Canvas fingerprints. This randomization is to eliminate the potential effect of people using similar devices at different times of the day when visiting our website (e.g. people tend to use mobile devices in the evening more than desktops). Afterwards for each Canvas fingerprinting sample, we picked 90% of the fingerprints for each sample and calculated the entropy for this 90%. We repeated this 10 times for each of the 23 Canvas samples we gathered. The 90% happens to be 471 or 472. While doing this, we ensured that a single fingerprint value gathered from a Canvas sample coming from a user visit is included exactly 9 times. This is to ensure that all the values are represented enough in this assessment. Table 4.6 lists all the entropies calculated for each of the 10 90% folds, calculated for the 23 Canvas samples.

After calculating the entropy of the 10 folds of fingerprint values for each Canvas sample, we calculated several other relevant values: the mean of the 10 folds' entropy (to compare it with overall entropy of the entire sample of 524 visits), highest fold entropy, lowest fold entropy and the standard deviation. Table 4.7 summarizes these values for every Canvas sample, while Figure 4.3 presents this information in a plot with the Canvas sample number in the x-axis and entropy value in the y-axis. The values in the plot include the overall entropy of a single Canvas sample (based on the 524 user visits) represented by the blue diamond, the mean of the 10 folds' entropies represented by the orange circle, and the highest and lowest fold entropy represented by 2 "X" symbols. The closer the overall entropy to the mean entropy of the 10 folds, the more confidence we have in the reported Canvas sample entropy. If we look at the plot in Figure 4.3, we find that 3 of the Canvas samples (i.e. Canvas 15, 17 and 20) happen to have the overall entropy within 1 standard deviation around the mean entropy, giving more confidence of the reported results for these samples. One of these samples is our enhanced Canvas sample we propose for the best distinguishing (Sample 20). The plot also shows that 11 of the Canvas samples have the overall entropy within the highest and lowest fold entropy of the same Canvas, which shows high confidence that is relatively less than when the overall entropy is within the standard deviation from the mean entropy. These 11 Canvas samples are Canvas samples 1, 2, 10, 12, 14, 15, 16, 17, 20, 21 and 22. We dropped Canvas sample 2 from the plot due to its low entropy, to makes the plot more interpretable to the reader. The Canvas samples' printed images and their description are available in Appendix A.

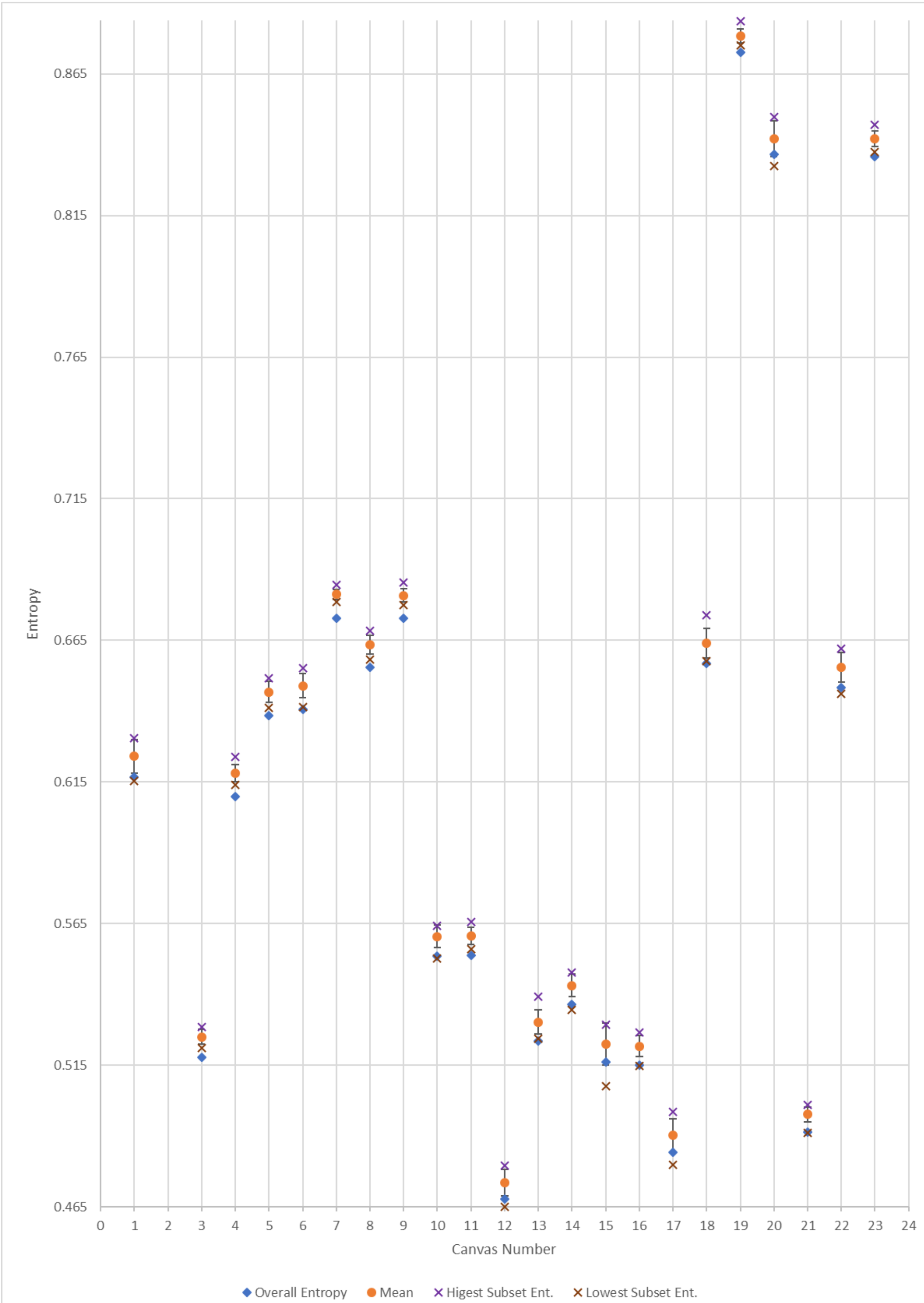


Figure 4.3: Overall vs. Mean Entropy

Canvas Sample	1st Fold	2nd Fold	3rd Fold	4th Fold	5th Fold	6th Fold	7th Fold	8th Fold	9th Fold	10th Fold
Can. 1	0.631	0.631	0.615	0.625	0.626	0.628	0.623	0.618	0.616	0.629
Can. 2	0.356	0.351	0.351	0.354	0.346	0.357	0.358	0.346	0.353	0.355
Can. 3	0.524	0.522	0.528	0.525	0.524	0.524	0.528	0.529	0.526	0.521
Can. 4	0.614	0.622	0.618	0.616	0.616	0.624	0.620	0.619	0.618	0.616
Can. 5	0.645	0.642	0.648	0.641	0.646	0.650	0.652	0.651	0.644	0.648
Can. 6	0.648	0.646	0.641	0.652	0.648	0.652	0.645	0.649	0.653	0.655
Can. 7	0.680	0.681	0.680	0.679	0.682	0.682	0.684	0.681	0.685	0.681
Can. 8	0.665	0.659	0.668	0.658	0.663	0.665	0.664	0.667	0.666	0.661
Can. 9	0.677	0.682	0.686	0.682	0.681	0.681	0.679	0.681	0.682	0.679
Can. 10	0.553	0.559	0.558	0.563	0.556	0.560	0.564	0.564	0.564	0.563
Can. 11	0.566	0.556	0.562	0.559	0.559	0.562	0.561	0.564	0.557	0.561
Can. 12	0.478	0.473	0.473	0.480	0.475	0.479	0.469	0.465	0.470	0.475
Can. 13	0.525	0.529	0.531	0.524	0.534	0.530	0.529	0.532	0.530	0.539
Can. 14	0.540	0.548	0.546	0.541	0.547	0.544	0.542	0.545	0.535	0.545
Can. 15	0.507	0.529	0.529	0.513	0.519	0.527	0.521	0.527	0.524	0.528
Can. 16	0.515	0.527	0.520	0.518	0.526	0.522	0.524	0.521	0.521	0.524
Can. 17	0.488	0.499	0.486	0.488	0.480	0.487	0.497	0.488	0.492	0.497
Can. 18	0.660	0.660	0.674	0.666	0.662	0.661	0.666	0.671	0.662	0.658
Can. 19	0.880	0.875	0.879	0.879	0.878	0.878	0.876	0.884	0.876	0.878
Can. 20	0.833	0.849	0.844	0.834	0.838	0.844	0.842	0.850	0.838	0.850
Can. 21	0.497	0.498	0.499	0.497	0.491	0.497	0.500	0.498	0.501	0.498
Can. 22	0.651	0.662	0.660	0.646	0.652	0.658	0.654	0.656	0.653	0.662
Can. 23	0.841	0.843	0.847	0.843	0.841	0.845	0.838	0.841	0.840	0.842

Table 4.6: Normalized Entropy of Random 90% Folds of Each Canvas Sample

Canvas Sample	Mean Ent. of Folds	Overall Entropy	Highest Fold Ent.	Lowest Fold Ent.	Std. Dev. (x10^{-3})	Mean - STD	Mean + STD
Can. 1	0.624	0.617	0.631	0.615	5.89	0.618	0.630
Can. 2	0.353	0.348	0.358	0.346	4.07	0.349	0.357
Can. 3	0.525	0.518	0.529	0.521	2.61	0.522	0.528
Can. 4	0.618	0.610	0.624	0.614	3.06	0.615	0.621
Can. 5	0.647	0.638	0.652	0.641	3.72	0.643	0.650
Can. 6	0.649	0.641	0.655	0.641	4.18	0.645	0.653
Can. 7	0.681	0.673	0.685	0.679	1.78	0.679	0.683
Can. 8	0.663	0.656	0.668	0.658	3.39	0.660	0.667
Can. 9	0.681	0.673	0.686	0.677	2.25	0.679	0.683
Can. 10	0.560	0.553	0.564	0.553	3.96	0.557	0.564
Can. 11	0.561	0.554	0.566	0.556	2.96	0.558	0.564
Can. 12	0.474	0.468	0.480	0.465	4.66	0.469	0.478
Can. 13	0.530	0.524	0.539	0.524	4.25	0.526	0.534
Can. 14	0.543	0.536	0.548	0.535	3.91	0.539	0.547
Can. 15	0.522	0.516	0.529	0.507	7.40	0.515	0.530
Can. 16	0.522	0.515	0.527	0.515	3.65	0.518	0.525
Can. 17	0.490	0.484	0.499	0.480	5.90	0.484	0.496
Can. 18	0.664	0.657	0.674	0.658	5.31	0.659	0.669
Can. 19	0.878	0.873	0.884	0.875	2.44	0.876	0.881
Can. 20	0.842	0.837	0.850	0.833	6.36	0.836	0.848
Can. 21	0.498	0.491	0.501	0.491	2.66	0.495	0.500
Can. 22	0.655	0.648	0.662	0.646	5.23	0.650	0.661
Can. 23	0.842	0.836	0.847	0.838	2.71	0.839	0.845

Table 4.7: Reliability of our Reported Canvas Samples' Normalized Entropy

Our enhanced Canvas sample is composed of a linear gradient with 5 color stops in the background, two stroke rectangles with round edges and different line widths, and two filled rectangles, one of which is transparent. All the shapes have shadows, different colors, positions and dimensions, and some of them overlapping. On top of that are all the alphabetical letters in capital and small, numbers, and 25 characters, printed twice in the Windows local fonts "Palatino Linotype" and "Arial" with a size of 25, in addition to 21 different selected emojis. Fig. 4.4 shows our enhanced Canvas sample.

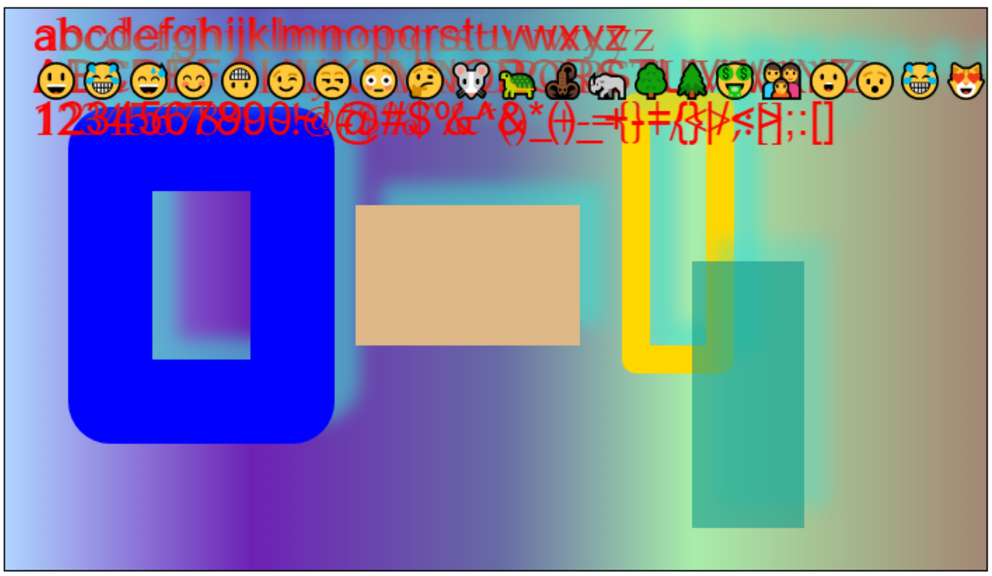


Figure 4.4: Our Enhanced Canvas Sample (Canvas 20)

4.5 General Observations

The following observations are drawn from the various Canvas samples we gathered in Appendix A and show the impact of changing some factors on the number of distinct fingerprint values (refer to Fig. 4.2 and Table 4.2 for better visualization). Utilizing these observations can help in improving the samples a website uses to fingerprint its users.

- Canvas sample 2 contains 4 shapes in black, whereas Canvas 3 have the same 4 shapes in 4 different colors. By changing the colors, the number of distinct fingerprint values increased from 36 to 60.

- Canvas sample 3 has a white background, by changing the background into a gradient with 5 colors as in Canvas 5, the number of distinct fingerprint values increased from 60 to 105.

- Canvas 9 is the same as Canvas 5, except that the shapes have shadows, which increased number of distinct values from 105 to 132.

- By printing all the alphabetical letters in capital and small, numbers, and 25 characters twice in different Windows available fonts and adding many Emojis, the number of distinct fingerprint values increased from 132 in Canvas 9 to 276 in Canvas 20.

- Canvas 21 is what several researchers use for fingerprinting the users. By increasing the size of the sample (i.e. pixels) and adding more distinct letters and characters in Canvas 22, the number of distinct fingerprint values increased from 79 to 135.

- Canvas 1 contains some text and an Emoji, which results in 119 distinct fingerprint values. By printing many emojis (even with no text at all) in Canvas 18, the number of distinct fingerprint values became 152.

- Increasing the font size results in more distinct values as in Canvas 10 with the font Arial size 18 and 96 distinct values versus Canvas 11 with the font Arial size 20 and 99 distinct values. Also applies to samples 12 and 13, using Sirin Stencil (a font that is imported Google Web Fonts server) with sizes 12 and 15, resulting in 83 and 88 distinct fingerprint values respectively.

- Canvas 19 and Canvas 20 have the same components, except that Canvas 19 uses a font that is imported Google Web Fonts server. Because of this, Canvas 19 resulted in 317 distinct values as opposed to 276 in Canvas 20. This increase in Canvas 19 is caused by the occasional failure in loading the font from the web, which would be misleading as a device can have different fingerprints at different user visits.

- Printing a picture in a Canvas, like in sample 23, can be as good as our suggested enhanced Canvas version (i.e. Canvas 20) which contains many other various components.

- Google Chrome and Opera showed to have the most similar fingerprints among the other browsers when installed on the same device. Most of the Canvas samples obtained gave the same fingerprint for both browsers on the same device. However, our enhanced sample (Canvas 20) was able to distinguish both browsers. Unlike earlier ones, this observation is based on over 60 submissions we performed on Google Cloud servers using the browsers: Internet Explorer, Google Chrome, Firefox and Opera. We changed the hardware specifications after submitting the form from the 4 browsers.

4.6 Why Canvas is a Reliable Fingerprint

When Mowery et al. discovered the possibility of using Canvas to fingerprint users in 2012, they mentioned several characteristics that make Canvas a desirable fingerprinting technique [3]. These characteristics include: consistency in getting the same Canvas fingerprint value from the same user device, the high entropy in the returned value, transparency to the user as there is no indication that they are being fingerprinted, being independent of other fingerprints, and finally being readily obtainable since there is no access permission required to be able to fingerprint users.

Laperdrix et al. constructed a 17-attribute fingerprinting technique, and studied the distinguishing capabilities of each attribute on a dataset of 118,934 fingerprinted users, which was collected through a website they published: AmIUnique.org [1]. Table 4.8 summarizes the Normalized Shannons Entropy for the top 4 gathered attributes through the AmIUnique.org website for all of their samples, and divided by desktop samples only then mobile samples only. We notice that the normalized entropy is very low for the list of plugins and fonts for mobile devices, indicating that it can be challenging to fingerprint mobile devices using the list of plugins, due to the absence of plugins in mobile devices, and the list of fonts, given that a considerable proportion of the mobile devices do not have flash activated, and the flash API is used to gather the list of fonts. With the increased adoption and use of mobile devices to surf the web, it is important for fingerprinting attributes to have high entropy for both desktop and mobile devices. User

Agent and Canvas remain on the top of the attributes that have similar and high entropy for both desktops and mobile devices.

Table 4.8: Normalized Shannons Entropy for the Top 4 AMIUNIQUES Attributes [1]

Attribute	All	Desktop	Mobile
List of plugins	0.656	0.718	0.081
User agent	0.580	0.550	0.741
List of fonts	0.497	0.548	0.033
Canvas	0.491	0.475	0.512

The reasons why we see high potential in Canvas is that the web application which is fingerprinting users have control over what to render on the clients machine. Therefore, enhancing the sample gathered would increase the entropy and consequently the distinguishing capability significantly, unlike fonts, plugins and user-agent, which web applications do not have control over. Looking at the number of distinct fingerprints obtained by our enhanced Canvas sample (i.e. Canvas sample 20) in our experiment in this chapter, the results are promising. The numbers of distinct fingerprint values achieved by each of the non-Canvas fingerprint we gathered are all way less than our enhanced Canvas sample. With this, it is possible to enhance the distinguishability of users by utilizing enhanced samples of Canvas.

4.7 Conclusion

In this chapter, we aimed to provide high entropy Canvas samples to be used for better fingerprinting, check how their entropy compare to Canvas and non-Canvas

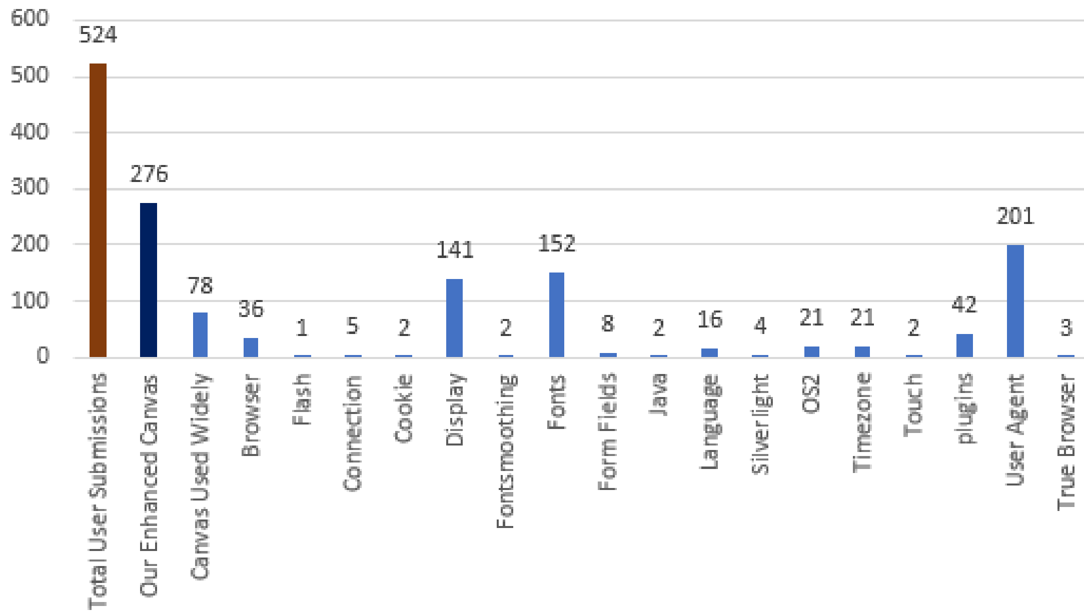


Figure 4.5: Number of Distinct Values per Non-Canvas Fingerprints

fingerprints, and come up with conclusions on how the different components added to a Canvas sample can boost the entropy. To achieve our targets we developed a website and hosted it publicly, asking users to enter details about the device from which they are visiting, while the website gathers 23 Canvas fingerprints and 17 non-Canvas fingerprints from each user visit. The website was distributed to our acquaintances, the academic community, social networks and paid visitors from Mechanical Turk, who provided 701 submissions, filtered down to 524 included in our study.

Our three topmost Canvas samples in entropy are higher than any Canvas and non-Canvas fingerprints in the literature reviewed in Section 2.2.2. The analysis show that the more content and characteristics a Canvas sample includes, the more capable it is in distinguishing web users. Our proposed enhanced version has all alphabetical characters written in several fonts, over 20 emojis, several shapes, shadows, and a completely covered background, resulting a normalized entropy of 0.837. The output of this empirical study confirm our initial expectation that the entropy of Canvas fingerprinting can be boosted by the web application that wishes to fingerprint users, as opposed to other types of fingerprints.

To assess the reliability of our reported findings, we compared the Shannon Entropy of a commonly used Canvas sample in our dataset versus AmIUnique, and the entropy in both dataset was almost identical, which gives confidence that our dataset is valid and representative of similar large dataset when calculating other samples' entropies. As another data validation measure, we calculated the entropy of ten 90% folds for each Canvas sample and compared the mean entropy resulted with the overall entropy of all the samples. The sample we propose in this chapter (sample 20) happened to have the overall entropy within one standard deviation from the mean entropy of the 10 folds. Finally, the Operating Systems percentages in our dataset is the closest to the actual Operating Systems market share when compared to the significant fingerprinting dataset in section 2.3, and hence it is the least in bias.

CHAPTER 5

CANVAS FINGERPRINTING FOR ATTACK DETECTION

This chapter visits two different applications that can significantly benefit from Canvas fingerprinting. These two applications are the detection of fake accounts on web applications and the prevention of session hijacking. In this chapter, we propose a methodology for both applications. In addition to that, an empirical study has been carried out to understand the effectiveness of the proposed technique in detecting fake accounts on web applications, and a proof of concept was built to test the usage of Canvas fingerprinting in session hijacking prevention.

5.1 Motivation

Detecting fake accounts on web applications and online social networks have several benefits including preventing the owners of these accounts to sell unauthentic "likes" and followers on these social networks. Moreover, the detection of fake accounts prevents attackers from creating many accounts for illegitimate purposes such as studying the session management of a website to find vulnerabilities that can be exploited. In addition to that, the results of behavior studies of the target audience of a website may be impacted, which may cause financial implications if the website operators took decisions depending on these studies. For these reasons and multiple other reasons, we proposed the use of Canvas fingerprint to detect the creation of fake accounts in websites and online social networks [2].

Cookies are being used for session authentication since mid-90's. When a user logs in to a web application and his credentials get validated, session cookies are generated and sent to the client's web browser. The web browser then attaches the cookie values in every subsequent request to prove that the user is authenticated [42], instead of the need for users to provide their credentials in every sensitive page they visit. Session hijacking happens when an unauthorized user steals a session that belongs to another user by obtaining the other user's cookies, which gives the stealer unauthorized access to information or services. Therefore, it became important to provide protection measures against session hijacking.

5.2 Proposed Methodology

The methodology we propose for our two security applications are composed of a two-stage process that starts with rendering and drawing some shapes or text on a Canvas after the HTML page is sent to the user's browser. Then, the website retrieves the pixel data back and stores it in the database to use it for identifying the user or the session depending on the application. Choosing Canvas fingerprinting for the two applications comes from the consistency in the pixel data retrieved from the Canvas across different user visits from the same device, in addition to the high entropy which helps in differentiating users (refer to Section 4.6). The Canvas fingerprint is linked to the user account in the fake accounts detection application. Therefore, if the number of accounts with the same Canvas fingerprint is large, flags should be raised and investigated as they may indicate that the owner of these accounts is the same person. On the other hand, the Canvas fingerprint is linked to the session of the user in the session hijacking prevention application. So, if the value of this fingerprint is changed while the same session is active, it means the users session has been stolen, because the Canvas fingerprint is more persistent than the user's session. The proposed process for both applications has two stages: Canvas storing and Canvas checking. The Canvas storing stage is similar in the two applications, while the checking stage takes different approaches based on the application. The first stage of the fake accounts detection process is implemented in the registration HTML page. The registration web page has a Canvas HTML element <canvas> with shapes or text rendered on it in the client machine. When the user signs up for a new account and his input gets validated, the JavaScript function **toDataURL()** retrieves the Base64 representation of the pixel data. This data is hashed and stored in the database associated with the username of the newly created account. A brief overview of the storing stage is summarized in Figure 5.1. The first stage of the session hijacking prevention process is implemented in the login page and the Canvas pixel data is retrieved the same way when the user logs in and used as a fingerprint for the user session.

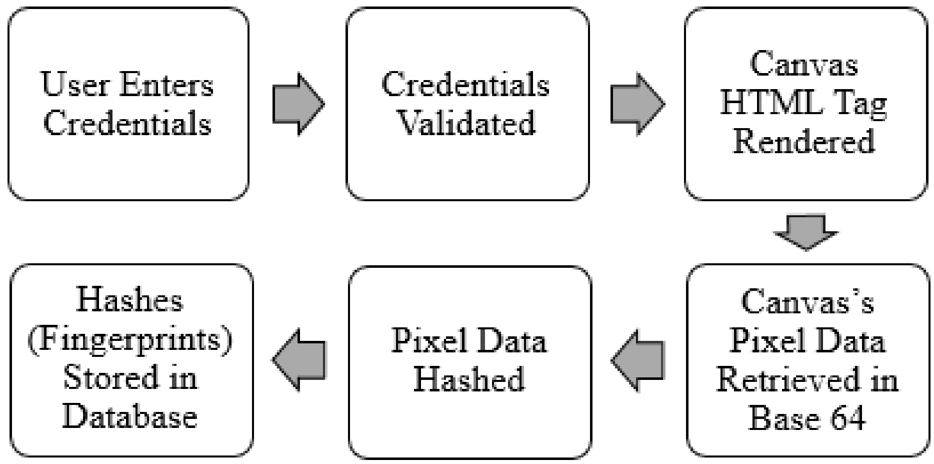


Figure 5.1: Canvas Storing Stage of the Detection Process [2]

The second stage of the fake account detection process checks for repeated fingerprints in the database that are associated with different accounts at the time of accounts creation. When the number of accounts associated with a single fingerprint exceeds a specific threshold, the operators of this website or online social network should perform further investigation as this gives an indication that these accounts may be fake accounts that belong to the same user, and that they could be used to perform malicious activities. The threshold should be specified taking into account statistical analyses to reach an optimal value that reduces false positives and false negatives. The authors of Beauty and the Beast published the website AmIUnique.org to analyze 17 attributes that can be used to fingerprint users, including Canvas fingerprint [1]. The analysis showed that the 118,934 responses have 8,375 distinct Canvas values, with an average of around

14 sample per Canvas value. This study and similar ones would help in deciding the best thresholds. After the analysis and investigations, the operator can take the appropriate actions such as disabling or blacklisting the accounts. Figure 5.2 covers the Canvas repetition checking stage of the detection process.

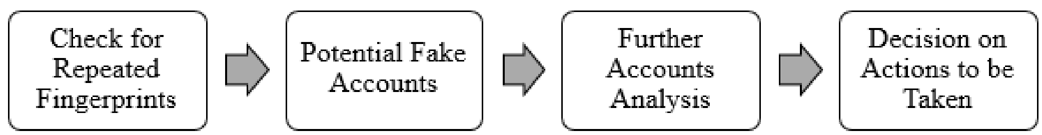


Figure 5.2: Repetition Checking Stage of the Fake Accounts Detection Process [2]

The second stage of the session hijacking prevention process checks if the Canvas value associated with the session which was stored in the first stage is still the same. This stage is deployed in any webpages the web applications owners would like to secure against session hijacking. When the user navigates to a sensitive page in which the Canvas checking function is implemented, the web application retrieves the pixel data resulted from rendering the same Canvas and hashes it using the same hash function used in the first stage. After that, the Canvas fingerprint newly generated in the second stage is compared to the fingerprint stored earlier in the database during the first stage of the process. If the newly generated fingerprint matches the stored one, it means that the current session belongs to the same legitimate user who successfully logged in earlier, and the user will be able to access and stay at the same page she was requesting. If the result is a mismatch, then there is a very high probability that the user's session has been hijacked and the user will be redirected to the login page to re-enter her credentials, and if correctly entered, the value of the new fingerprint is stored in the database as in the Canvas storing stage that was done at the login time. The implementation of this process can accommodate users who login from different devices at the same time by allowing the web application to store more than one session ID per user with their associated Canvas fingerprint for each device. Figure 5.3 illustrates the second stage of the session hijacking prevention process.

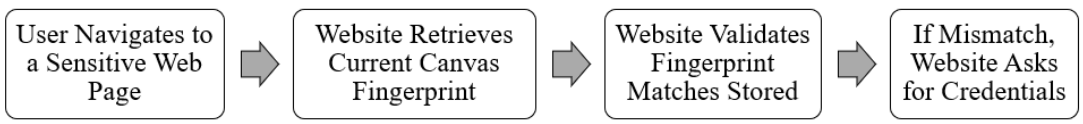


Figure 5.3: Checking Stage of the Session Hijacking Prevention Process

The target of the two-stage processes is protecting web applications from the creation of fake account and users' sessions from getting hijacked. Hence, it is not required to gather user identifying information that can be considered as a privacy breach. This is why we are using the fingerprint as a black-box and only store the hash of the pixel data we get back from the Canvas. The hashing is expected to introduce collisions, which can be taken into consideration by the person performing the analysis in the second stage of the fake account detection process. The collisions are not very critical in the session hijacking application either since what we care about is detecting if the person using the session is not the owner of the account, as opposed to having unique identifiers for users.

5.3 Empirical Study of the Fake Account Detection Application

A simple website with user registration functionality was built and hosted publicly. PHP was used as the server side programming language, and JavaScript was the client side scripting language. Figure 5.4 shows how the website looks like. We selected 14 different participants to take part of this empirical study. For the accessibility of the research participants, we selected university students that were attending an Information Security Fundamentals course, who were also aware of the purpose of the experiment beforehand. It was not a requirement to have participants that are privacy or security aware. Any participant with basic skills on browsing the web would be sufficient. Our selection of the privacy aware participants has impacted our reported false negatives rate. After that, we asked the participants to register on the website a random number of times with different usernames and passwords each time. The total number of registered accounts was 148 account. We then tried to use the participants' input to detect which group of accounts belong to the same user by following the two-stage process described earlier. To validate the output of our analysis of Canvases to determine if Canvas fingerprints belong to the same user, we asked the users to provide their email addresses every time they sign up for a new account.

The experiment showed that the proposed detection process is effective in detecting which accounts belong to the same user with a good accuracy. Table 5.1 points out the expected flags that should be raised when a fake account is

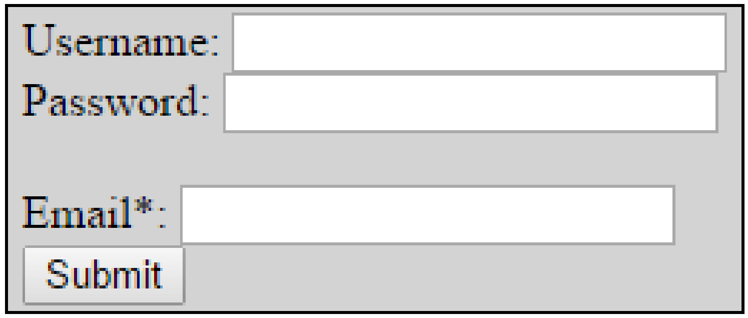


Figure 5.4: Simple Website with Registration Functionality

created, and compares it with actual raised flags. A flag is raised when a Canvas fingerprint being stored in the database matches at least one value previously stored.

A total of 120 flags were raised in the repetition checking stage. 8 of these flags were false positive. These cases occurred because of machines with very similar or identical hardware and software specifications. In one of the cases, a collision occurred by two desktop machines next to each other in a lab. The checking stage did not alert for 9 flags. 8 of these flags were missed because the users were changing the browser's configurations by changing the zoom ratio. When investigated, we found out that this was due to some participants using Mozilla Firefox, who were intentionally trying to register for multiple account while being undetected. This is an impact of our selection for the participants of our empirical study, which could have been avoided if we selected random web users. With further analysis and replication of their activity, we saw that in some cases the zoom ratio had impact on the fingerprints coming from Mozilla Firefox. We were not able to replicate this behavior on any other browser. One missed flag was because Internet Explorer 8 does not support Canvas. The Canvas sample used in this experiment is sample 1 in Appendix A. Using a sample with a better entropy (like Sample 20) could have decreased the false positives. It is noteworthy that this study included 148 registered accounts that were created by 14 participant, which might not adequate to represent the rate of false positives and false negatives that would be observed if the study covered the entire population of web users. However, this study shows initial findings on the effectiveness of the proposed technique, and could be expanded to cover significantly larger audience for more accurate results.

Table 5.1: Expected and Actual Results [2]

Registered Accounts	148
Expected Flags	121
Raised Flags	120
Falsely Raised Flags (False Positives)	8
Missed Flags (False Negatives)	9
False Positives Inaccuracy	6.67%
False Negatives Inaccuracy	7.44%

5.4 Proof of Concept for the Session Hijacking Prevention Application

As a proof of concept, we developed a website using PHP as the server side language, and JavaScript as the client side scripting language. This website has basic login functionality that enables the user who is successfully logged in to post comments. This website is vulnerable to session hijacking, meaning that by copying the cookie value that is created to store the session to another browser in the same device or a different device, the session is obtained without the need to provide the username and password.

To protect against stealing sessions, we implemented the proposed solution by building two extra functions: Store Canvas and Check Canvas. When a user enters the credentials and logs in successfully, the Store Canvas function renders some text on the Canvas, retrieves pixel data, and then calculates the fingerprint and stores it in the database associated with the current session. The Check Canvas function is included in every sensitive page in the website, so that whenever the user who has the current session visits a sensitive web page, the Check Canvas function renders the same text on the Canvas, retrieves pixel data, calculates the fingerprint, and compares it to the fingerprint stored earlier in the database. In the case of a mismatch, the user is sent back to the login page to provide the credentials since there is a high probability that the session has been stolen.

We tested the website after adding the Store Canvas and Check Canvas functions. When the user logged in to a browser, a new cookie was created to store the session, and the user was able to post comments to the website. When the cookie value was copied to another browser in the same device or in a different device, and tried to post a comment, the web application detected that the Canvas fingerprint associated to the session has changed and redirected the user to the login page to enter the credentials. When entering the correct credentials from the other browser, the fingerprint stored in the database was updated with the new fingerprint value.

5.5 Existing Techniques to Detect Fake Accounts and Prevent Session Hijacking

Several researchers approached the problem of detecting the creation of fake accounts from different angles. Some researchers tried to detect the creation of bulk of fake accounts by finding patterns in the username of the account and patterns in the time of certain activities performed by the account owner, such as posting or tweeting in specific and fixed time patterns [43]. Other researchers analyze the mutual friends between accounts and use this to indicate whether some accounts are fake [44]. These detection techniques can be circumvented by the creator of the fake accounts. For example, the creation can happen in random times to avoid the fixed time patterns, and the usernames can be randomized in order not to have detectable naming patterns. The technique proposed in this research does not rely on finding time and naming patterns, nor does it rely on mutual friends analysis. Thus, it can be implemented in addition to the existing techniques to achieve layered security.

In an effort to prevent session hijacking, the authors of [45] presented a client side mechanism named SessionShield, which protects users against session hijacking even if the web application is vulnerable. The idea of SessionShield works by detecting and isolating session identifiers from the browser, as this information is not used by legitimate client side scripts. Another research proposes getting rid of cookies completely and deploying One Time Cookies (OTC). OTC is resistant to session hijacking, and keeps the simplicity and performance advantages of conventional cookies [42]. The technique proposed in this research is a server side protection measure that can be employed easily in a web application along with other server side and client side techniques without needing to deploy a whole new cookies system and without compromising performance.

5.6 Conclusion

Canvas fingerprinting is typically used by web applications to track users, but it can also be utilized for the good. In this chapter we proposed and explored two novel constructive use cases. We also studied the effectiveness of these use cases.

The first use case we propose helps detecting the creation of fake accounts on social networks. This protects the web application owners from unneeded behaviors such as the selling of unauthentic followers, and impacting the target audience analyses performed by the web application owners. For this we propose a two stage process that starts with retrieving and storing the Canvas fingerprint at the time of account creation, and linking it to the user created. The second stage checks for repeated fingerprints and raises alerts for potential fake accounts. The empirical study we performed to validate this technique raised 120 flags where 121 flags were expected, resulting in 6.67% false positives and 7.44% false negatives.

The second use case prevents session hijacking. To implement this use case, the Canvas fingerprint is linked to the user session, since the fingerprint is more persistent than the session. the web application checks for the fingerprint in the sensitive web pages. If the fingerprint is different while the same session is active, this indicates a stolen session and the application asks the user to re-enter their credentials. We developed a proof of concept that proved the effectiveness of this use case.

CHAPTER 6

ATTACKING PRIVACY USING CANVAS FINGERPRINTING

The purpose of this chapter of our research is to show specific examples of how web applications could utilize the easily obtainable Canvas fingerprints to de-anonymize web users. We present here five different targets of de-anonymizing users, and how they can be achieved via Canvas.

1. Linking guest user to a logged in user
2. Conclude a person or entity owns several accounts, or that several users are related (e.g. family members, roommates, friends or classmates)
3. Conclude that a user owns several devices
4. Attribute activities done across different web applications to the same user even when using different browsers on the same or different devices
5. Confirmation of identity speculation

This chapter also explains how the Tor browser protects users from being fingerprinted via Canvas, and sheds the light on some functionality failures Tor users face when using Canvas applications.

6.1 Linking a guest user to a logged in user

Let us say a website W would like to identify its guest users and link them to registered user accounts. This would typically be done easily via cookies. However, a privacy aware user who would like to protect their identity can erase the cookies. Website W can utilize Canvas to identify its visitors as in Fig. 6.1 and as demonstrated in these steps:

1. If a Visitor V1 visits website W and performs some activities as a guest user (even if the visitor doesnt own an account yet), the website will link the activity to the visitors fingerprint F1.

2. If in the future a Visitor V2 visits the same website W while being logged in as user U with a fingerprint F2 that is equal to F1, the website can attribute the activities done by the guest user to the user U, and can conclude that V1 = V2.

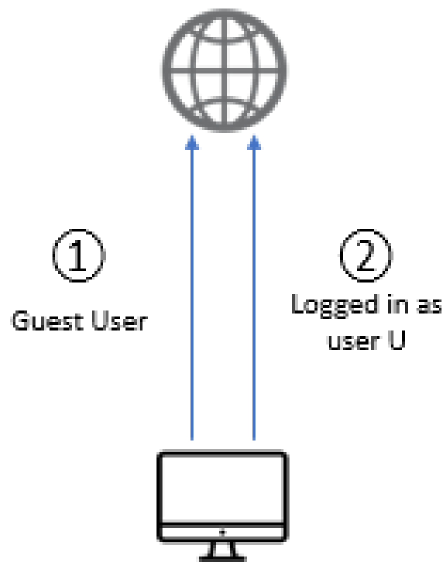


Figure 6.1: Linking a guest user to a logged in user

6.2 Conclude a person or entity owns several accounts

Lets say a website W uses Canvas to identify its visitors as in Fig. 6.2 and the following steps:

1. If a Visitor V1 visits the website W while being logged in as user U1, the website will link the user U1 to the obtained fingerprint F1.

2. If in the future a Visitor V2 visits the same website W while being logged in as user U2 whose fingerprint F2 is equal to F1, the website can understand that the owner of U1 and U2 is the same person, or that the visitors V1 and V2 are related (e.g. family members, roommates, friends or classmates)

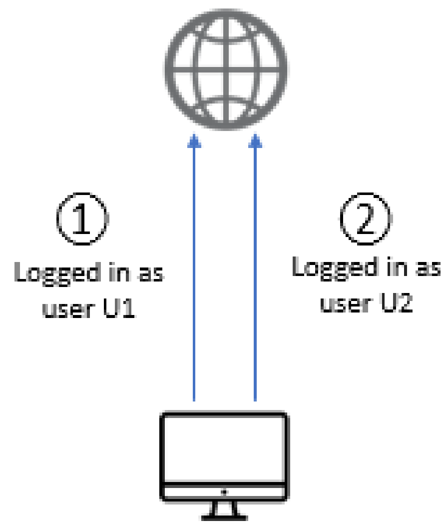


Figure 6.2: Conclude a person or entity owns several accounts

6.3 Conclude that a user owns several devices

Lets say a website W uses Canvas to identify its visitors as in Fig. 6.3 and the following steps:

1. If a Visitor visits the website W and performs some activities while being logged in as user U, the website will link the fingerprint F1 to the User U, and can as well record the type of activities done while the fingerprint is F1.

2. If in the future a Visitor visits the same website W while being logged in as the same user U with a fingerprint F2 that is different from F1, and performs different types of activities from the first visit, the website can know that the user U owns two different devices and understand which types of activities the user performs from each.

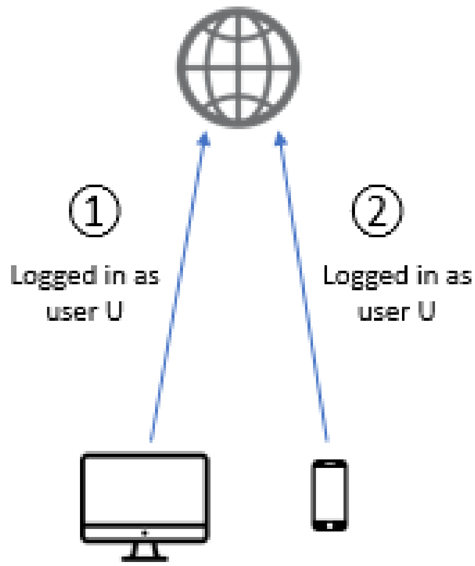


Figure 6.3: Conclude a user owns several devices

6.4 Attribute activities done across different web applications to the same user despite using different devices

Let us say websites W1 and W2 use Canvas to identify their visitors, and are owned by the same entity or have mutual agreement to share users fingerprints (Note that W1 and W2 can be the same website). W1 and W2 can utilize the steps in Fig. 6.4 and described as follows:

1. If a visitor V1 visits the website W1 without being logged, the website will link the activities done by V1 to the obtained fingerprint F1.

2. If V1 visits W2 while being logged as user U, W2 will link the User U to the obtained Fingerprint F1.

3. Once a visitor V2 logs in to W2 as user U, W2 will link the newly obtained fingerprint F2 to the same user U. Therefore, F1 and F2 potentially belong to the same person.

4. Given that W1 and W2 share the fingerprints information, if V2 visits W1 without being logged in, W1 will be able to attribute the activities done by V2 to the same source as V1.

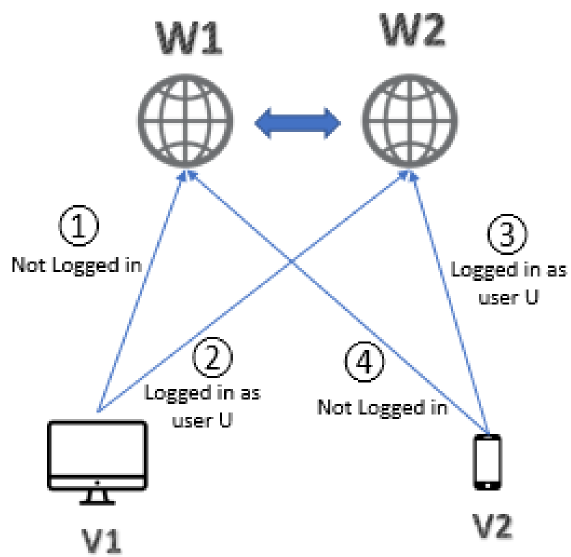


Figure 6.4: Attribute Activities to Users Across Devices

6.5 Confirmation of identity speculation

Lets say websites W1 and W2 constructed their own banks of Canvas fingerprints that link each of their registered users to the fingerprints of the devices from which they regularly visit the websites, and assuming that the websites are owned by the same entity or have mutual agreement to share users fingerprints. An identity speculation can be confirmed as in Fig. 6.5 and the following steps:

- User U1 has an account on website W1, and speaks up about sensitive matters using a pseudonym.

- User U2 has an account on website W2, and has provided his accurate identifying information like his name and email address.

- One of the websites (or a governmental entity) suspects that user U1 is the same as user U2 (e.g. because they use the same style of writing), and wishes to confirm this speculation.

- As users U1 and U2 visit the websites W1 and W2 respectively from multiple devices, W1 would populate its bank of fingerprints by linking U1 to the fingerprints of the devices used in the visits, and W2 would populate its bank the same way.

- By comparing the fingerprints of the devices associated to the two users in each of the fingerprint banks, it is possible to confirm the speculation if some fingerprints match, and the confidence would increase when more matches are found.

Note that W1 and W2 can be the same website, and that the same approach still applies to users visiting the websites from the same device using browsers with different fingerprints.

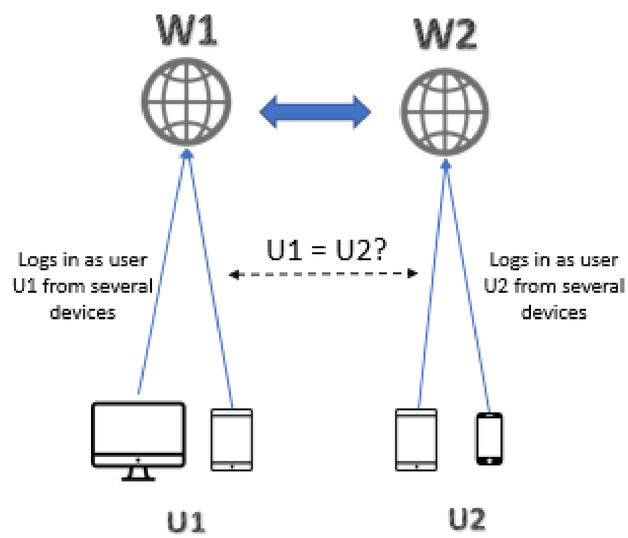


Figure 6.5: Confirmation of identity speculation

6.6 Privacy of Tor Browser Users

A user of Tor is protected from getting fingerprinted via Canvas on at least two layers: the scripting language level, and the specific implementation Tor for Canvas. At the scripting language level, the user of Tor has the option of stopping any scripts from running on the browser by enabling the option “Forbid Scripts Globally”, which is what Tor recommends, and what a typical privacy aware user would choose. If this option is enabled, the user experience browsing the web would not be smooth, because many dynamic web pages will not function properly with the scripting language disabled. In our case, all the Canvas applications and games failed when using this option since Canvas requires a scripting language. Tor browser has an effective mechanism that informs the user when a website is trying to obtain the Canvas fingerprint by looking for calls to the function **toDataURL()** and displaying the pop up in Fig. 6.6 if any is found. The user has the following three options to select from:

- "Not Now"
- "Never for this site"
- "Allow in the future"

With the first option being the default and the second being the recommended option. Tor does not allow the fingerprinting to happen unless the user approves it.

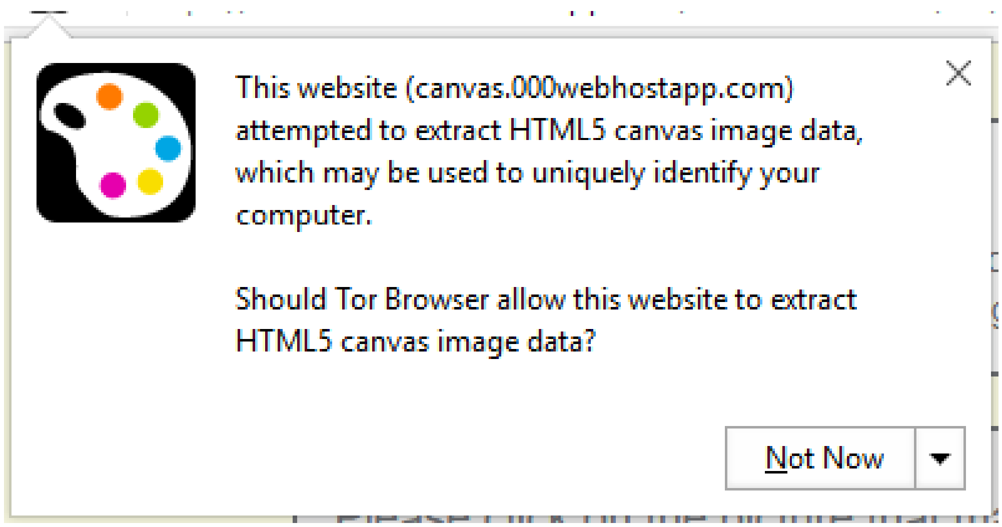


Figure 6.6: Canvas Fingerprint Alert in Tor

Many Canvas applications work fine with Tor as long as the fingerprinting algorithm is not used. If the function **toDataURL()** is a requirement in implementing a Canvas application (like for saving the resulted picture from a Canvas Paint application), the application works fine until the user tries to store the picture of the Canvas, then Tor would display the Canvas fingerprinting alert asking for users approval. Fig. 6.7 shows this scenario in action, where we only got the fingerprinting alert at the time of saving the picture. In some Cases, we observed failure in the functionality of some Canvas applications when using Tor, while the applications work fine in other major browsers. Fig. 6.8 shows an example of a Paint application that failed to store the drawing when browsing via Tor, while the application works fine on other browsers.

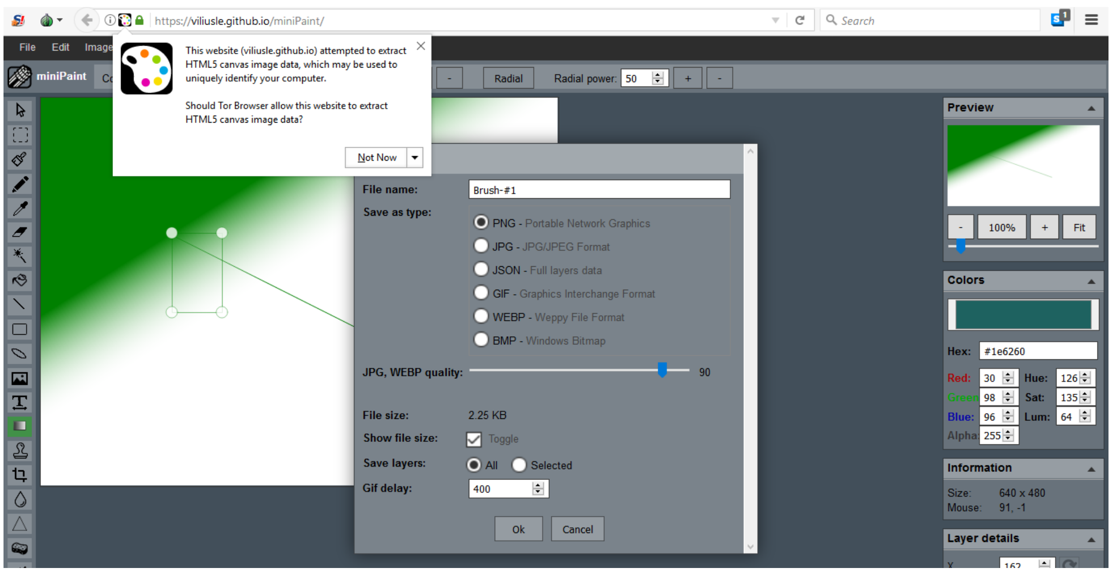


Figure 6.7: Paint Application Based on Canvas Only Displaying the Alert When Saving the Drawing

While studying how applications function on Tor versus other browsers, the alert Tor displays gave us insights on some applications that may appear to be using Canvas only for legitimate purposes, while in fact it uses Canvas for tracking users as well. Fig. 6.9 is for a game that allowed us to play and at a certain point the alert was triggered, which indicates the site is trying to fingerprint users. Not allowing the fingerprinting did not prevent us from continuing playing.

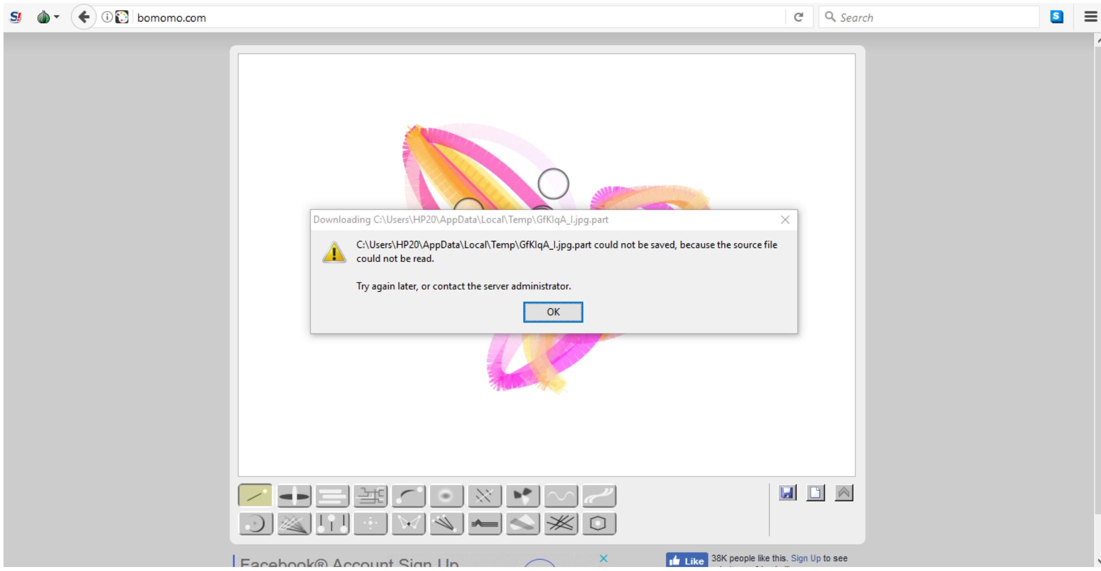


Figure 6.8: Canvas Paint Application Failing to Store the Drawing in Tor

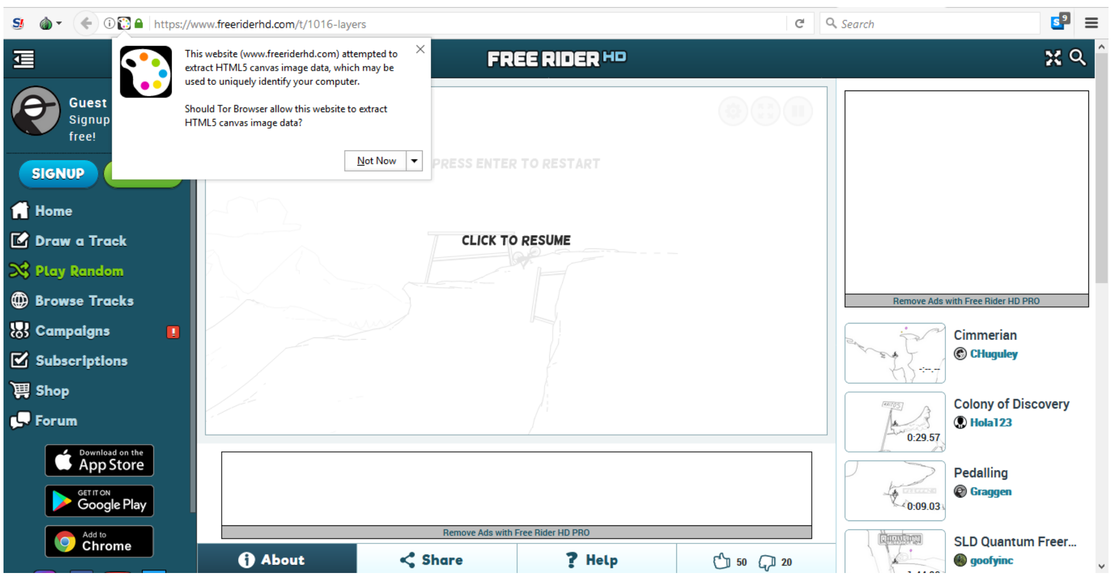


Figure 6.9: Canvas Game Trying to Fingerprint Users

6.7 Conclusion

In this chapter, we discussed five specific scenarios where Canvas fingerprints can be utilized by web applications to de-anonymize web users. Some of these scenarios -if not all- may have been implemented by web applications to track the users across web applications, even when using different devices. Our research provides more accuracy and effectiveness to these applications that attack privacy especially when the tracking web applications utilize our enhanced Canvas sample, or follow the guidelines we provide in Chapter 4 to create their own Canvas sample. It is noteworthy that there are companies specialized in cross-device tracking using a combination of several techniques [28]. These companies store graphs that link users to their devices, and can provide this information to other parties or web applications which subscribe to the service and may also be sharing information back with the tracking companies.

This chapter also demonstrates that while Tor browser provides effective mechanisms to protect web users from being fingerprinted, these mechanisms do not affect the importance and relevance of performing research on Canvas for several reasons, including the inconvenience caused for normal web users in terms of functionality failure as well as the large number of alerts that a user needs to respond to. Moreover, considering the portion of web users who are using Tor on regular basis, they account for a negligible percentage of those whom a web application owner wishes to track. According to StatCounter, the top 29 used web browsers account for over 99.8% of the market share as of October 2018. These 29 browsers do not include Tor among them [46].

CHAPTER 7

RESEARCH VALIDATION, REPLICATION, CONCLUSION AND FUTURE WORK

This chapter explains how we validated our performed empirical studies and points out the threats we see to the validity of our research in Section 7.1, while Section 7.2 gives the research community the key elements with which they can replicate our research to confirm our findings, or to pick up from where we reached with our empirical studies instead of reinventing the wheel. Finally, we conclude this thesis and suggest future direction. The sections of this chapter refer mainly to the four major experiments and empirical studies we carried out:

1. Canvas prevalence on the web

2. Canvas distinguishing capability analysis

3. Detecting the creation of fake accounts on web applications

4. Session hijacking prevention on web applications

7.1 Research Validation and Threats to Validity

When studying the Canvas prevalence in Chapter 3.2 to show the cost of disabling Canvas, our initial results showed that only 2.45% websites use Canvas. However, when validating this by comparing with the literature, we found that a research published in 2014 showed that over 5.5% of the top 100,000 Alexa websites used Canvas for tracking purposes [4], and this percentage kept growing till it reached 10.44% in April 2016 when measured on the top 10,000 Alexa websites [32], which cannot be more than the number of websites using Canvas in general for all purposes. After investigations, we found out that our scrapping algorithm misses some cases with which Canvas can be included in a website. Our revised algorithm showed that 24.04% of websites use Canvas as of January 2018.

We reported in Chapter 3.2 that 21.96% of the top 10,000 Alexa websites were detected by our crawling script to be using Canvas for all purposes, and estimated that 11.52% of them (over 50% of the reported) use Canvas for legitimate purposes. This estimation has three threats to validity. First, one year and nine months separate the execution of our crawling script and the crawling done by [32], which may have impacted the percentage of deployed Canvas for fingerprinting purposes on the web. Second, it is possible that the reported percentage of websites using Canvas fingerprinting in [32] was subject to false negatives especially with their false positive elimination criteria, and consequently affecting their reported results. Third, since the estimation of the of the Canvas used for legitimate purposes is a direct subtraction of the overall usage minus tracking usage, it considers a website that uses Canvas for both legitimate and tracking purposes only in the tracking side, which is a more restrictive (hence safer) assumption that may only reduce the prevalence for legitimate usage. These three threats are only for the estimated prevalence for legitimate purposes, while the overall percentage we reported does not change.

In addition to the Canvas samples we designed to fingerprint the users in the Canvas distinguishing capability analysis performed in Chapter 4, we also gathered the Canvas sample used widely (like in [1] and [4]) to confirm the quality of our data by comparing the entropy we get in our dataset with the other large datasets including AmIUnique [1] when using the same fingerprinting sample. The entorpy obtained by the Canvas sample in both datasets is almost identical, showing the reliability of our dataset. We also gathered 17 non-Canvas fingerprints to study the effectiveness of Canvas against each fingerprint, showing that Canvas is the highest in entopy and distinguishing capability. A threat to the validity of this empirical study is that we based our analysis on 701 user responses (524 after filtration), as opposed to 118,934 in the research performed by Laperdrix et al. [1], and the potential bias the distribution of our dataset, as the different versions of Windows constructed 56% of the responses, and the submissions done from the Chrome browser were 70% of the dataset.

The empirical study performed in Chapter 5 aimed to asses the effectiveness of the technique we proposed for detecting the creation of fake accounts on web applications using Canvas. The study was based on 148 registered accounts created by 14 different people. These numbers may not be sufficient to accurately represent the false positives and negatives when it comes to the entire population of web users, but they give an indication to the effectiveness of the technique.

Chapter 6 discusses five scenarios of how to attack the privacy of web users and de-anonymize them. In these scenarios the attacks are shown in diagrams and explained theoretically, but the effectiveness is not empirically proven yet because this is not in the scope of our research. As a future direction, it is valuable to design empirical studies that show the effectiveness of these scenarios, and assess the potential overhead.

7.2 Replication and Reproducing Results

In the Canvas prevalence empirical study of our research, we showed that the percentage of websites including Canvas exceeds 24%, which proved the high cost of disabling Canvas at the browser level. In order for researchers to see how our findings may change over the time, we created a repository with the source code we used to reach our results, which takes a CSV file (with the list of URLs to be visited) as an input and outputs a subset list of URLs for the websites containing Canvas. The repository can be found on: **https://github.com/abouollo/Canvas-Prevalence**.

In the Canvas distinguishing capability empirical study, we gathered and studied the distinguishing capability of 23 Canvas samples to fingerprint users. We summarized the the characteristics of each sample in Table 4.2 and provided the output of how the samples look like when rendered using Google Chrome on an HP 2000 Notebook that is running Microsoft Windows 10 and has Intel HD Graphics 4000 as its graphics card in Appendix A. In addition to that, we created a repository with the source code we used to draw each of the 23 samples, and gave directions on how to use the code in a different website if someone wishes to replicate our research or extend it. The repository can be found on: **https://github.com/abouollo/Canvas-Samples**.

In the empirical study we performed to validate the technique we proposed for detecting the creation of fake accounts on web applications, we used Canvas sample number 1 on the Canvas Samples Github repository **https://github.com/abouollo/Canvas-Samples**, however according to the distinguishing capability analysis performed in Chapter 4, we expect Canvas sample number 20 to give better results (i.e. less false positives and negatives).

For the proof of concept built for the technique we proposed to prevent session hijacking on web applications, we used sample number 1 on the Canvas Samples Github repository. We also expect Canvas sample number 20 to give more effective results.

That said, We see that there is a large room for enhancement in the Canvas samples that can be used in tracking users, and in addition the two novel constructive applications we proposed for utilizing Canvas, it is possible to brainstorm and study the effectiveness of more applications. The repositories we provide can help as a starting point.

7.3 Conclusion

In this thesis we explored Canvas, one of the most powerful HTML features from various perspective, looking at its history, support, and the functionality for which Canvas was introduced, showing why it is a serious candidate to replace Flash which is being discontinued for many security concerns. We demonstrated how JavaScript is used to manipulate Canvas, and showed the API functions being utilized for the tracking purposes, how they empower web applications developers, and why these functions cannot be easily avoidable, which is the main basis of this thesis.

We implemented a script that uses Selenium to scrape the top 100,000 Alexa websites looking for the websites that contain Canvas element in the main source HTML, any iframe, and any JavaScript component that is loaded with the landing page of the websites. The result showed that the Canvas is very widespread, as over 24% of the websites use this feature, and hence disabling it to protect users' privacy comes with a large cost.

To understand the distinguishing capability of Canvas, we performed an empirical study that included over 500 participants who provided their Canvas and non-Canvas fingerprints. The main targets of this study are to compare the Canvas samples to other widely used fingerprinting techniques, drafting guidelines on how to enhance the Canvas samples, and to provide strong Canvas samples in terms of distinguishing users. The results demonstrate how the various components and characteristics of each of the 23 different Canvas samples gathered impact the distinguishing capability of the sample. The enhanced Canvas sample we proposed surpassed every other Canvas and non-Canvas fingerprint studied by AmIUnique [1] and Panopticlick [21], and was able to raise the distinguishing entropy from 0.49 to 0.83.

Moreover, we proposed two novel applications where Canvas fingerprint can be constructively utilized to detect and prevent attacks. The first application is to detect fake accounts creation on web applications through a two-stage process, starting by storing the Canvas fingerprint at the time of account creation, followed by a repetition checking stage to analyze and raise alerts for potential fake accounts. To assess the effectiveness of our proposed methodology, we performed an empirical study that included 148 accounts registered on a website we built. The analysis resulted in 6.67% of false positives inaccuracy and 7.44% of false negatives inaccuracy. The second application is a technique for session hijacking prevention. We built a proof of concept to demonstrate how a vulnerable application can be protected from session hijacking by our suggested mechanism. The key benefit here is that these two techniques can be implemented in addition to any existing techniques to achieve layered security.

This work also explores scenarios of how web applications can take advantage of

Canvas fingerprint to attack the privacy and de-anonymize web users who could be browsing through different devices and visiting separate web applications. These scenarios included linking a guest user to a logged in user, concluding a person or entity owns several accounts, identifying that several users are related (e.g. family members, roommates, friends or classmates), concluding that a user owns several devices, attributing activities done across different web applications to the same user even when using different browsers on the same or different devices, and confirming identity speculation.

7.4 Future Work

In the Canvas prevalence area, to overcome the threats to validity and to achieve more accurate results, it is beneficial to replicate our Canvas prevalence experiment for all purposes as well as replicate the Canvas prevalence only for tracking purposes at the same time frame to avoid any potential impact of time in the trends of the ever-changing web. To avoid the false positives we expect in our research findings, it is important pay more attention to the obfuscation techniques and find methods to overcome them. It can be useful to record the websites that uses Canvas constructively and for tracking in the same web page.

For the Canvas distinguishing capability analysis, our research and analysis were based on the fingerprinting data from 524 user visits. Although this dataset gives an understanding of how the distinguishing capability of Canvas is increased with the enhanced samples (especially with the normalized entropy), it is important to confirm the findings on a much larger dataset for more reliable results, while trying to maintain less biased samples in terms of operating system and web browser market share. The observations we found can also be utilized for designing new Canvas samples that are even more in entropy.

In addition to that, we suggest selecting some of the most distinguishing Canvas samples and identify what the actual differences in the rendered pixels are. This can be done by creating an animation that cycles fast through all the images resulted from the dataset, or by constructing a big matrix where we subtract pairs of images and highlight the pixels that differ. Accomplishing this could reveal the underlying causes of the differences and lead to further principles.

Further more, we suggest deep investigations to identify the root causes of the differences in pixel rendering. As starting points, we recommend to record and analyze the system calls triggered by the rendering of different Canvas samples (e.g. one sample can be an empty Canvas while another may have some simple component, or two samples the vary in complexity). System calls are the way via which an application requests services from the kernel of the operating system. By analyzing the differences in the type and the amount of triggered system calls when rendering different Canvas samples, it may be possible to identify the main factors contributing to the pixel differences. This can help quantify the extent to which each hardware components like CPU, GPU and graphics card or software configurations like operating systems and browser types and versions contribute to the overall differences in the rendered Canvas.

We proposed the fake account creation detection application via Canvas, and tested the effectiveness based on 148 registered accounts created by 14 participants. This experiment can be repeated on a larger scale of audience to confirm the effectiveness. It is also important to study how a malicious user can defeat our proposed technique (by methods like Canvas URL Data randomization) and implement defences against that.

We also proposed and implemented a session hijacking prevention technique utilizing Canvas. This implementation we provided only accommodates an account login from a single browser at a time. This implementation can be enhanced to allow simultaneous legitimate logins while continuing to detect session hijacking. Calculating the overhead of this technique and quantifying the dissatisfaction that may be caused by this protection measure are open research areas to consider in future research.

It would be valuable to address the destructive use cases of Canvas, as the attack scenarios are shown in diagrams and explained theoretically. Designing and performing empirical studies can better assess the effectiveness of these scenarios, and measure the potential overhead.

Finally, utilizing the GitHub repositories we provide would help in replicating all the empirical studies performed in this research, and is going to provide more trustworthiness to our findings.

REFERENCES

[1] P. Laperdrix, W. Rudametkin, and B. Baudry, "Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 878–894.

[2] A. Abouollo and S. Almuhammadi, "Detecting malicious user accounts using canvas fingerprint," in *Information and Communication Systems (ICICS), 2017 8th International Conference on*. IEEE, 2017, pp. 358–361.

[3] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in html5," *Proceedings of W2SP*, pp. 1–12, 2012.

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

[5] A. Gómez-Boix, P. Laperdrix, and B. Baudry, "Hiding in the crowd: an analysis of the effectiveness of browser fingerprinting at large scale," in *WWW 2018: The 2018 Web Conference*, 2018.

[6] D. Geary, *Core HTML5 canvas: graphics, animation, and game development.* Pearson Education, 2012.

[7] T. Bujlow, V. Carela-Español, J. Sole-Pareta, and P. Barlet-Ros, "A survey on web tracking: Mechanisms, implications, and defenses," *Proceedings of the IEEE*, vol. 105, no. 8, pp. 1476–1510, 2017.

[8] E. Mills, "Device identication in online banking is privacy threat, expert says." https://www.cnet.com/news/device-identification-in-online-banking-is-privacy-threat-expert-says/, April 2009, [Online; posted 24-April-2009].

[9] S. Macbeth, "Tracking the trackers: Analysing the global tracking landscape with ghostrank," Technical report, Ghostery, Tech. Rep., 2017.

[10] G. Team, "Tracking the trackers." [Online]. Available: https://www.ghostery.com/lp/study/

[11] P. E. Ketelaar and M. van Balen, "The smartphone as your follower: The role of smartphone literacy in the relation between privacy concerns, attitude and behaviour towards phone-embedded tracking," *Computers in Human Behavior*, vol. 78, pp. 174–182, 2018.

[12] A. Hannak, G. Soeller, D. Lazer, A. Mislove, and C. Wilson, "Measuring price discrimination and steering on e-commerce web sites," in *Proceedings of the 2014 conference on internet measurement conference.* ACM, 2014, pp. 305–318.

[13] J. Mikians, L. Gyarmati, V. Erramilli, and N. Laoutaris, "Detecting price and search discrimination on the internet," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks.* acm, 2012, pp. 79–84.

[14] ——, "Crowd-assisted search for price discrimination in e-commerce: First results," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies.* acm, 2013, pp. 1–6.

[15] K. Lobosco, "Facebook friends could change your credit score," *CNN, https://money.cnn.com/2013/08/26/technology/social/facebook-credit-score/index.html?hpt=hp_t2*, 2013.

[16] "Very personal finance," Jun 2012. [Online]. Available: https://www.economist.com/finance-and-economics/2012/06/02/very-personal-finance

[17] "How the nsa piggy-backs on third-party trackers," Nov 2013. [Online]. Available: http://cyberlaw.stanford.edu/publications/how-nsa-piggy-backs-third-party-trackers

[18] T.-F. Yen, Y. Xie, F. Yu, R. P. Yu, and M. Abadi, "Host fingerprinting and tracking on the web: Privacy and security implications." in *NDSS*, 2012.

[19] S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. W. Felten, "Cookies that give you away: The surveillance implications of web tracking," in *Proceedings of the 24th International Conference on World Wide Web.* ACM, 2015, pp. 289–299.

[20] A. Soltani, S. Canty, Q. Mayo, L. Thomas, and C. J. Hoofnagle, "Flash cookies and privacy." in *AAAI spring symposium: intelligent information privacy management*, vol. 2010, 2010, pp. 158–163.

[21] P. Eckersley, "How unique is your web browser?" in *Privacy Enhancing Technologies*, vol. 6205. Springer, 2010, pp. 1–18.

[22] R. Upathilake, Y. Li, and A. Matrawy, "A classification of web browser fingerprinting techniques," in *New Technologies, Mobility and Security (NTMS), 2015 7th International Conference on*. IEEE, 2015, pp. 1–5.

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

[24] K. Boda, Á. M. Földes, G. G. Gulyás, and S. Imre, "User tracking on the web via cross-browser fingerprinting," in *Nordic Conference on Secure IT Systems*. Springer, 2011, pp. 31–46.

[25] "Google code archive - long-term storage for google code project hosting." [Online]. Available: https://code.google.com/p/sputniktests/

[26] M. Mulazzani, P. Reschl, M. Huber, M. Leithner, S. Schrittwieser, E. Weippl, and F. Wien, "Fast and reliable browser identification with javascript engine fingerprinting," in *Web 2.0 Workshop on Security and Privacy (W2SP)*, vol. 5. Citeseer, 2013.

[27] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*. ACM, 2011, pp. 103–114.

[28] J. Brookman, P. Rouge, A. Alva, and C. Yeung, "Cross-device tracking: Measurement and disclosures," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 2, pp. 133–148, 2017.

[29] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh, "Mobile device identification via sensor fingerprinting," *arXiv preprint arXiv:1408.1416*, 2014.

[30] F. Alaca and P. C. van Oorschot, "Device fingerprinting for augmenting web authentication: classification and analysis of methods," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*. ACM, 2016, pp. 289–301.

[31] A. Narayanan, "Online tracking: A 1-million-site measurement and analysis," 2017.

[32] H. Le, F. Fallace, and P. Barlet-Ros, "Towards accurate detection of obfuscated web tracking," in *Measurement and Networking (M&N), 2017 IEEE International Workshop on*. IEEE, 2017, pp. 1–6.

[33] T. Groß and T. Müller, "Protecting javascript apps from code analysis," in *Proceedings of the 4th Workshop on Security in Highly Connected IT Systems*. ACM, 2017, pp. 1–6.

[34] "Javascript obfuscator." [Online]. Available: http://javascriptobfuscator.com/

[35] "Javascript obfuscate and encoder." [Online]. Available: http://www.jsobfuscate.com/

[36] "Panopticlick." [Online]. Available: https://panopticlick.eff.org/

[37] "Am i unique?" [Online]. Available: https://amiunique.org/

[38] "Operating system market share worldwide." [Online]. Available: http://gs.statcounter.com/os-market-share

[39] A. Corporate, "Flash the future of interactive content," https://theblog.adobe.com/adobe-flash-update/, 2017.

[40] R. Data, "W3schools online web tutorials," *Online Document available on: https://www.w3schools.com /html/html5*$_c$*anvas.asp*$, Lastvisited$: $23rdJuly$, 2008.

[41] "Human intelligence through an apiaccess a global, on-demand, 24x7 workforce." [Online]. Available: https://www.mturk.com/

[42] I. Dacosta, S. Chakradeo, M. Ahamad, and P. Traynor, "One-time cookies: Preventing session hijacking attacks with stateless authentication tokens," *ACM Transactions on Internet Technology (TOIT)*, vol. 12, no. 1, p. 1, 2012.

[43] S. Gurajala, J. S. White, B. Hudson, and J. N. Matthews, "Fake twitter accounts: profile characteristics obtained using an activity-based pattern de-

tection approach," in *Proceedings of the 2015 International Conference on Social Media & Society*. ACM, 2015, p. 9.

[44] L. Jin, H. Takabi, and J. B. Joshi, "Towards active detection of identity clone attacks on online social networks," in *Proceedings of the first ACM conference on Data and application security and privacy*. ACM, 2011, pp. 27–38.

[45] N. Nikiforakis, W. Meert, Y. Younan, M. Johns, and W. Joosen, "Sessionshield: Lightweight protection against session hijacking." *ESSoS*, vol. 11, pp. 87–100, 2011.

[46] "Browser market share worldwide." [Online]. Available: http://gs.statcounter.com/browser-market-share

[47] C. J. Hawthorn, K. P. Weber, and R. E. Scholten, "Littrow configuration tunable external cavity diode laser with fixed direction output beam," *Review of Scientific Instruments*, vol. 72, no. 12, pp. 4477–4479, December 2001. [Online]. Available: http://link.aip.org/link/?RSI/72/4477/1

[48] A. S. Arnold, J. S. Wilson, and M. G. Boshier, "A simple extended-cavity diode laser," *Review of Scientific Instruments*, vol. 69, no. 3, pp. 1236–1239, March 1998. [Online]. Available: http://link.aip.org/link/?RSI/69/1236/1

[49] C. E. Wieman and L. Hollberg, "Using diode lasers for atomic physics," *Review of Scientific Instruments*, vol. 62, no. 1, pp. 1–20, January 1991. [Online]. Available: http://link.aip.org/link/?RSI/62/1/1

[50] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, “Aiding the detection of fake accounts in large scale social online services,” in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 2012, pp. 15–15.

[51] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham, “Fingerprinting information in javascript implementations,” *Proceedings of W2SP*, vol. 2, pp. 180–193, 2011.

[52] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna, “Towards detecting compromised accounts on social networks,” *IEEE Transactions on Dependable and Secure Computing*, 2015.

[53] K. Lee, J. Caverlee, and S. Webb, “Uncovering social spammers: social honeypots+ machine learning,” in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2010, pp. 435–442.

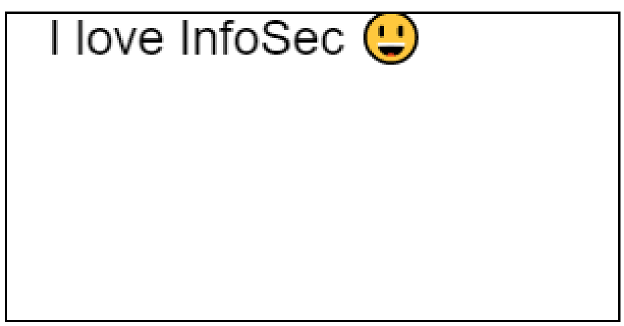
[54] C. Xiao, D. M. Freeman, and T. Hwa, “Detecting clusters of fake accounts in online social networks,” in *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*. ACM, 2015, pp. 91–101.

APPENDIX A

CANVAS SAMPLES

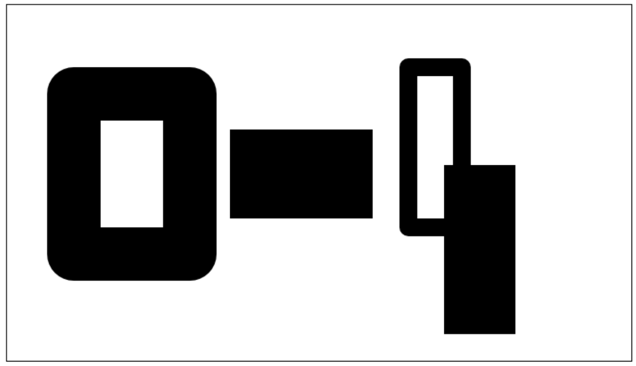
The screenshots were taken from Google Chrome (64bit) 62.0.3202.94 on Microsoft Windows 10 Home Single Language (Version 10.0.15063 Build 15063). The laptop model is HP 2000 Notebook PC. Graphics Card: Intel HD Graphics 4000.

Canvas Sample 1:



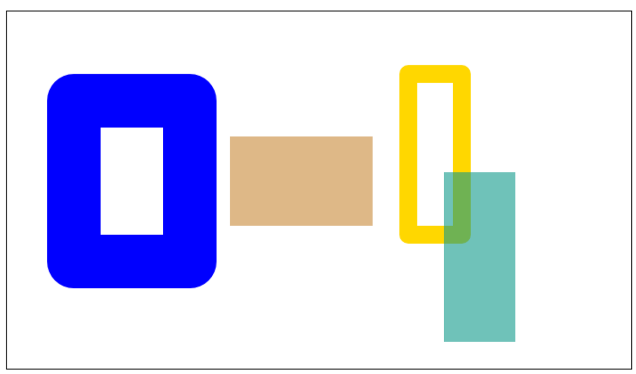
Contains the sentence ”I love InfoSec” written in the font “Arial” with size 18 pt, followed by an emoji.

Canvas Sample 2:



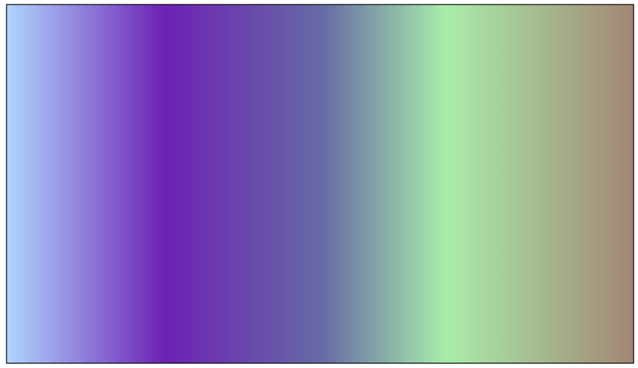
Two stroke rectangles (with round edges and different line widths) and two filled rectangles. All the shapes are in black, have different positions and dimensions, and some of them overlapping.

Canvas Sample 3:



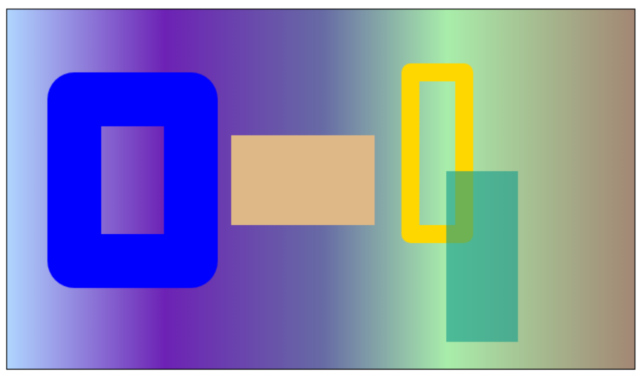
Two stroke rectangles with round edges and different line widths, and two filled rectangles, one of which is transparent. All the shapes have different colors, positions and dimensions, and some of them overlapping.

Canvas Sample 4:



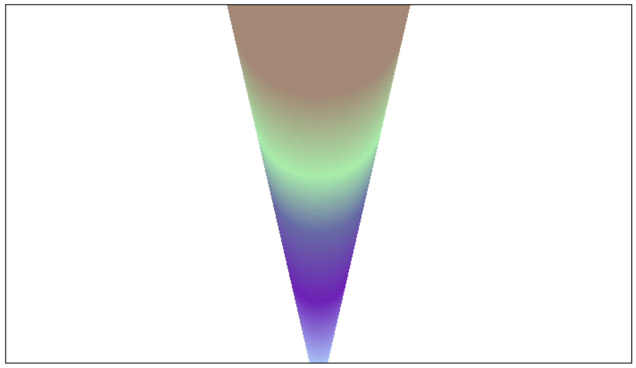
Linear gradient with 5 color stops.

Canvas Sample 5:



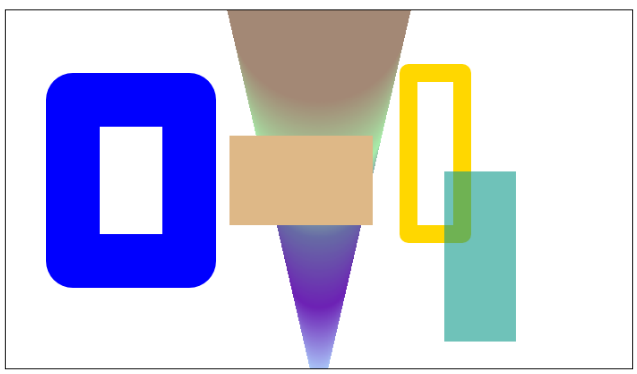
Linear gradient with 5 color stops in the background. Two stroke rectangles with round edges and different line widths, and two filled rectangles, one of which is transparent. All the shapes have different colors, positions and dimensions, and some of them overlapping.

Canvas Sample 6:



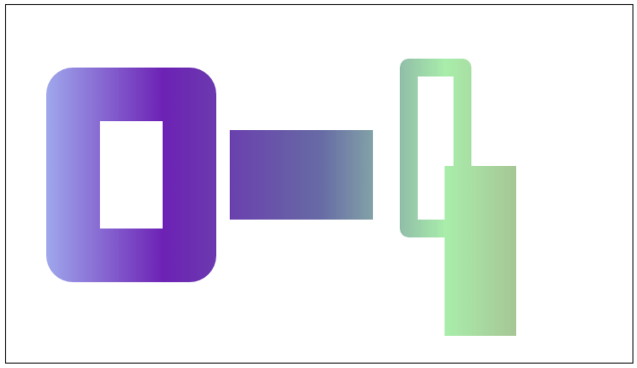
Radial gradient with 5 color stops in the background.

Canvas Sample 7:



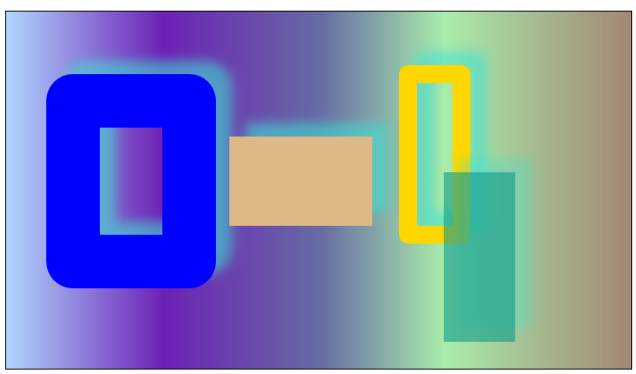
Radial gradient with 5 color stops in the background. Two stroke rectangles with round edges and different line widths, and two filled rectangles, one of which is transparent. All the shapes have different colors, positions and dimensions, and some of them overlapping.

Canvas Sample 8:



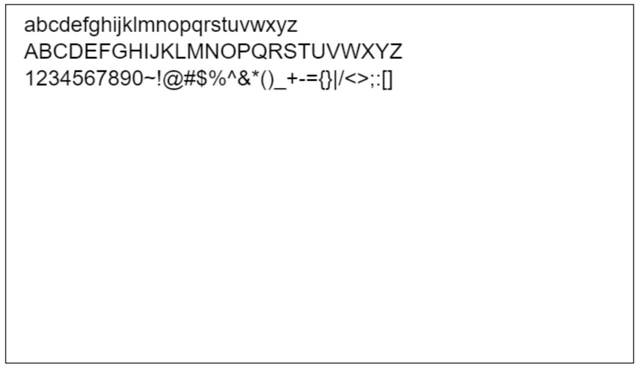
Two stroke rectangles with round edges and different line widths, and two filled rectangles. All the shapes have different positions and dimensions, and some of them overlapping. The shapes are colored with a gradient of 5 color stops.

Canvas Sample 9:



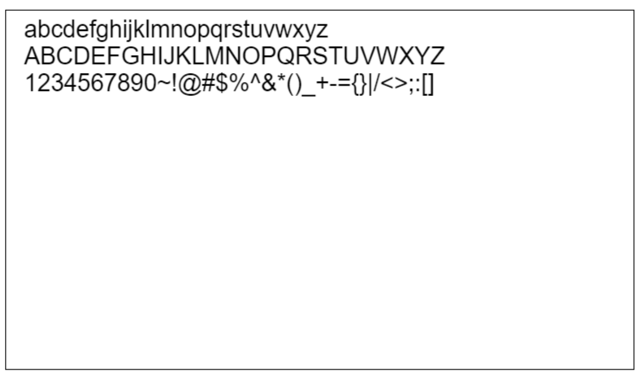
Linear gradient with 5 color stops in the background. Two stroke rectangles with round edges and different line widths, and two filled rectangles, one of which is transparent. All the shapes have shadows, different colors, positions and dimensions, and some of them overlapping.

Canvas Sample 10:



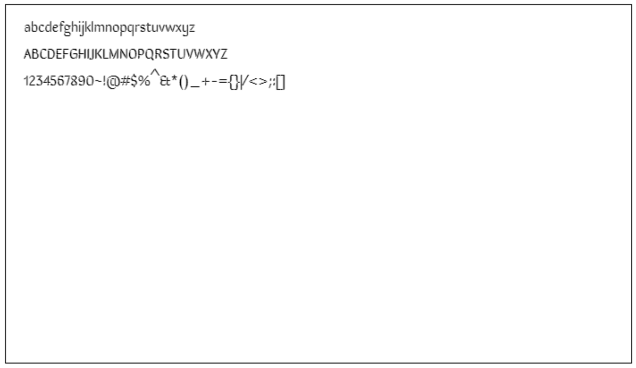
All the alphabetical letters in capital and small, numbers, and 25 characters, printed in the font “Arial” with a size of 18. This sample is to mimic a sample in [3].

Canvas Sample 11:



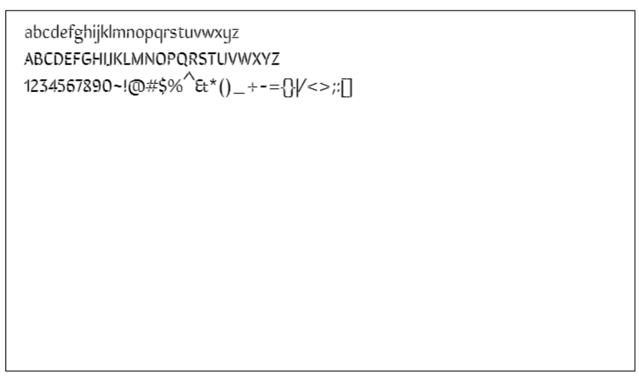
All the alphabetical letters in capital and small, numbers, and 25 characters, printed in the font “Arial” with a size of 20. This sample is to mimic a sample in [3].

Canvas Sample 12:



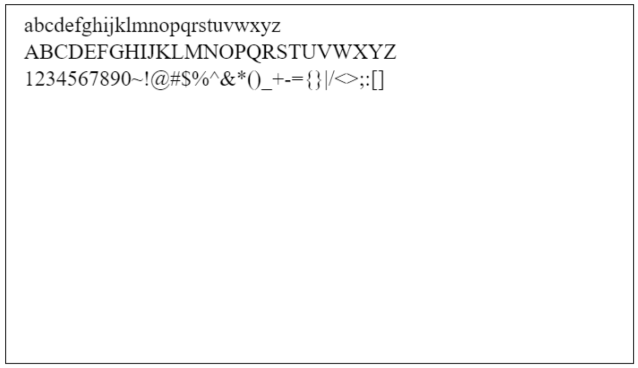
All the alphabetical letters in capital and small, numbers, and 25 characters, printed in the font "Sirin Stencil" from the Google Web Fonts server with a size of 12. This sample is to mimic a sample in [3].

Canvas Sample 13:



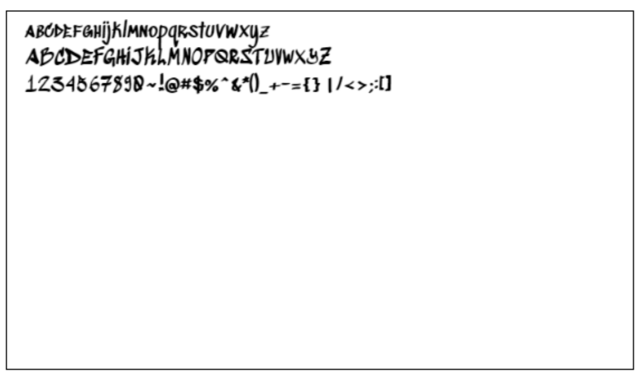
All the alphabetical letters in capital and small, numbers, and 25 characters, printed in the font "Sirin Stencil" from the Google Web Fonts server with a size of 15. This sample is to mimic a sample in [3].

Canvas Sample 14:



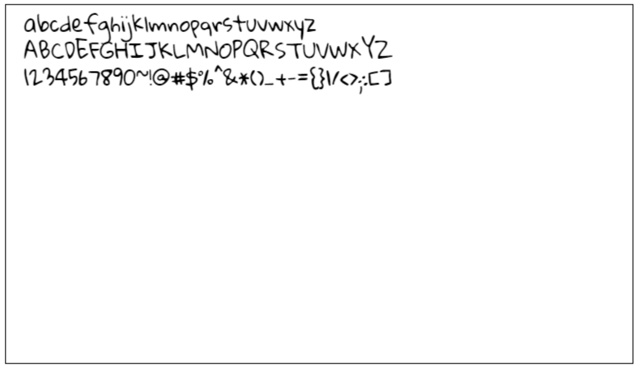
All the alphabetical letters in capital and small, numbers, and 25 characters, with a non-existent specified font "Fake-Font-Name" with a size of 18. This sample is to mimic a sample in [3].

Canvas Sample 15:



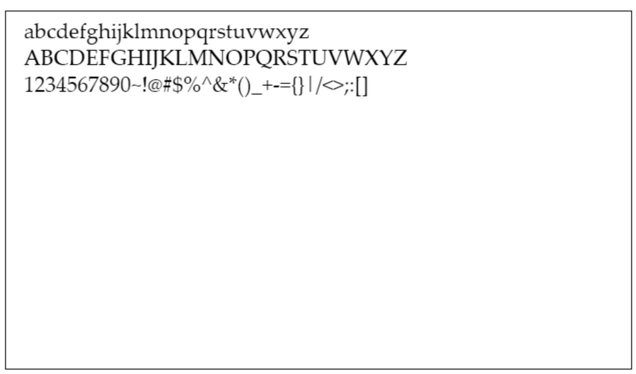
All the alphabetical letters in capital and small, numbers, and 25 characters, printed in the font "Sedgwick Ave Display" from the Google Web Fonts server with a size of 18.

Canvas Sample 16:



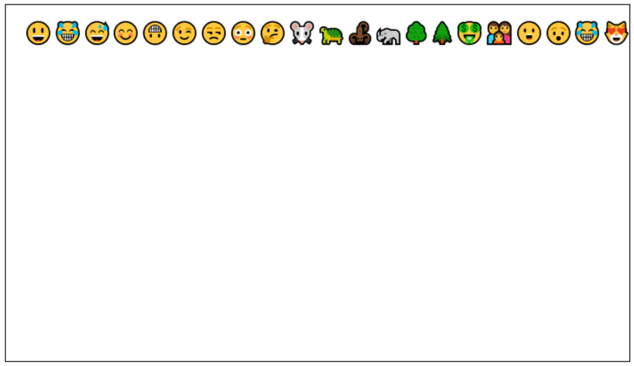
All the alphabetical letters in capital and small, numbers, and 25 characters, printed in the font "Gloria Hallelujah" from the Google Web Fonts server with a size of 18.

Canvas Sample 17:



All the alphabetical letters in capital and small, numbers, and 25 characters, printed in the local windows font "Palatino Linotype" with a size of 18.

Canvas Sample 18:

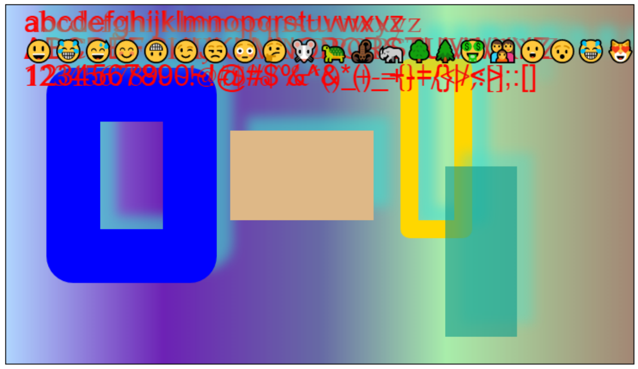


21 different selected emojis.

Canvas Sample 19:

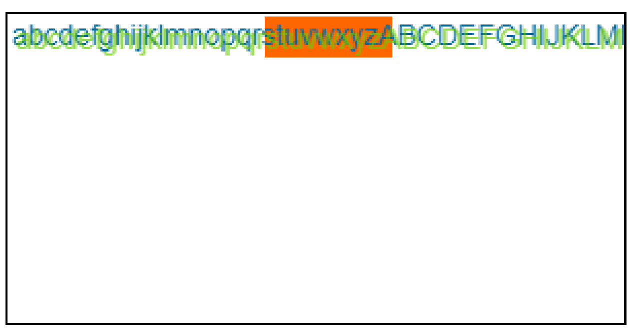
Linear gradient with 5 color stops in the background. Stroke rectangles and filled rectangles, one of which is transparent. All the shapes have shadows, different colors, positions and dimensions, and some of them are overlapping. On top of that are all the alphabetical letters in capital and small, numbers, and 25 characters, printed twice in the fonts “Palatino Linotype” (a Windows font) and “Sedgwick Ave Display” from Google Web Fonts with a size of 25. In addition to 21 different selected emojis.

Canvas Sample 20:



Linear gradient with 5 color stops in the background. Two stroke rectangles with round edges and different line widths, and two filled rectangles, one of which is transparent. All shapes have shadows, different colors, positions and dimensions, some of them overlapping. On top of that are all the alphabetical letters in capital and small, numbers, 25 characters, printed twice in the Windows local fonts “Palatino Linotype” and “Arial” with a size of 25, and 21 different emojis.

Canvas Sample 21:



All the alphabetical letters in small and many of them in capital printed overlapping in blue and green, with a little orange rectangle in the background. This sample is similar in concept to the ones presented in [4] and [1].

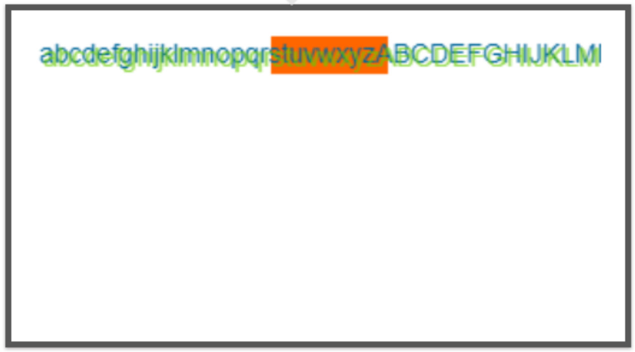
Canvas Sample 22:

All the alphabetical letters in capital and small, numbers, and 30 characters printed overlapping in blue and green, with a little orange rectangle in the background. This sample is similar in concept to the ones in [4] and [1].

Canvas Sample 23:

A landscape picture printed on a Canvas.

Canvas Widely Used:



This image was re-generated from the Canvas DataURL stored by the multi-factor algorithm in Table 4.1, using an online converter. It is similar to our Canvas sample 21.

Vitae

- Name: Ahmed Abouollo
- Nationality: Egyptian (Saudi Mother)
- Date of Birth: November 29, 1992
- Email: *a.abouollo@gmail.com*
- Permanent Address: Alkhobar, Saudi Arabia
- Previous Education: Bachelor of Science (B.S.) in Software Engineering, from King Fahd University of Petroleum and Minerals
- Research Interest:
  - Web Privacy & Tracking
  - Digital Forensics
- Practical Experience:
  - Cyber Defense
  - Security Operation Centers (SOC) Establishment
  - Digital Identity