Browser fingerprinting with Canvas

Проект по „ Основи на сигурното уеб програмиране “, 2018 г.

Факултет математика и информатика, Софийски университет

Изготвили:

45085, Димитър Илиянов Димитров, mitko.bg.ss@gmail.com
Информатика, 4 курс, 1 поток, 1 група

Ръководител: Филип Петров

2018 г.

Съдържание

1 Какво е Canvas? [1]

Canvas или <canvas> е инструмент на HTML5, който позволява динамично рисуване**.** По точно <canvas> съдържа това, което потребителят да нарисува, а самото рисуване(генериране) се извършва чрез JavaScript. Интересното при <canvas> е, че той е вграден директно в HTML, което значи, че е част от DOM-а и може директно да достъпва други елементи от HTML DOM. Той дава много по-голямо разнообразие от колкото Flash. Ето и малък пример:

```
<!DOCTYPE HTML>
<html>
    <body>
        <canvas id="myCanvas" width='300' height='200'></canvas>
        <script>
        let c=document.getElementById("myCanvas");
        let container=c.getContext("2d");
        container.rect(20,20,200,150);
        container.strokeStyle = "#FF0000";
        container.stroke();
        </script>
    </body>
</html>
```

Резултата от този код е следният:

Фиг. 1

2 Какво е Browser Fingerprinting? [1]

Вземането(снемането) на отпечатъци от криминалисти има за цел да събере база от данни, в която вкарва отпечатъци и ги свързва с съответните лица, които притежават отпечатъка. Така по-късно по този отпечатък те могат да идентифицират притежателя на този отпечатък.

При Browser fingerprinting идеята е аналогична. Като идеята при браузърите е просто да се идентифицира, че един и същ човек извършва дадени дейности без да се интересува директно от данните на този човек. От каква информация се интересуват хората, които ни следят? Отговорът е прост – трафик. В интернет пространството информация за даден трафик на потребител е много по ценна от неговите имена, град и подобни. След като е събрана информация за трафика тя може да бъде продадена на рекламни агенции, за да могат те да насочат конкретни реклами, които след анализ на трафика, са устанавили, за подходящи.

3 Как се осъществява Browser fingerprinting с Canvas? [1]

Canvas fingerprinting започва от момента, в който даден уебсайт възложи на вашия браузър да изрисува <canvas> обект. Самият обект не е отпечатакът, той е просто инструмент в изграждането на вашия отпечатък.

В основата на този вид отпечатъци е именно същата идея като при hash функциите. Две очевидно различни неща, след хеширане може да изглеждат почти еднакви. Това е именно идеята на Canvas fingerprinting. Въпреки, че за човешкото око <canvas> рисува напълно еднакви обекти това съвсем не значи, че компютърът ги възприема за еднакви също .

3.1 С какво са уникални браузър отпечатъците? [2], [3]

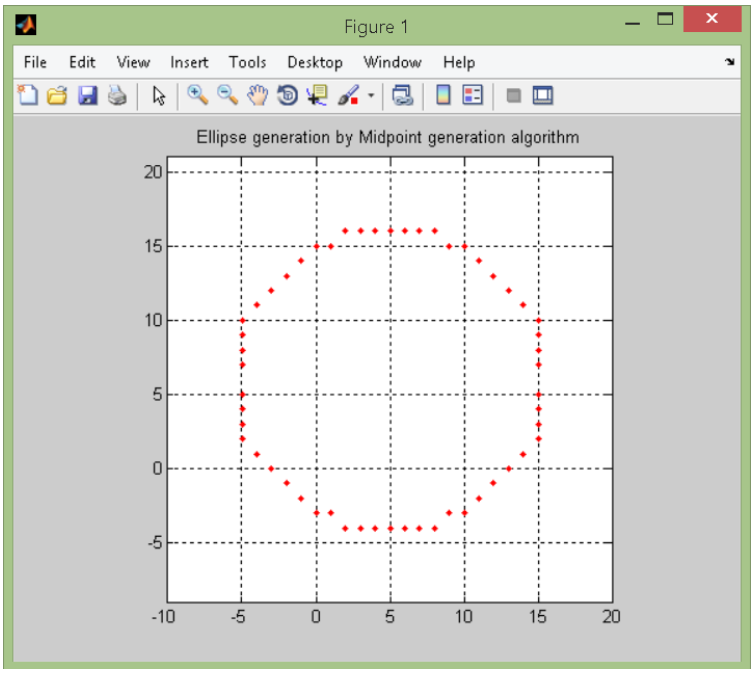
За да покажа с какво е уникален браузър отпечатъкът ще дам за пример една таблица:

Attribute	Value
User agent	Mozilla/5.0 (X11; Linux i686) Gecko/20100101 Firefox/25.0
HTTP accept	text/html,application/xhtml+xml,application/xml; q=0.9,*/*;q=0.8 gzip, deflate en-US,en;q=0.5
Plugins	Plugin 0: IcedTea-Web 1.4.1; Plugin 1: Shockwave Flash 11.2 r202
Fonts	Century Schoolbook, DejaVu Sans Mono, Bitstream Vera Serif, URW Palladio L, ...
HTTP DoNotTrack	1
Cookies enabled	Yes
Platform	Linux i686
OS (via Flash)	Linux 3.14.3-200.fc20.x86 32-bit
Screen resolution	1920x1080x24
Timezone	-480
DOM session storage	Yes
DOM local storage	Yes
I.E. User data	No

Тук е показан примерен браузър отпечатък. User agent и HTTP accept се пращат от HTTP headers към даден уебсайт. Останалата информация може да бъде взета много лесно чрез JavaScript – navigator & screen objects. Естествено има и други особености, които могат да влязат в отпечатъка като например JS функционалностите, които браузъра има(или няма). Още по интересно е, че може да се разбере и от хардуера и софтуера, като се гледа времето за рендърване на <canvas> обекта.

3.2 Какво представлява canvas fingerprint? [2], [3]

Преди да си отговорим на този въпрос, първо как по точно се рисуват тези елементи с <canvas>? За тези, които са учили компютърна графика би трябвало да е ясно, че това са математически формули, чрез които се изчислява как да се запълнят дадени пиксели в определеното пространство. Това, което се получава директно от математическата формула, е много суров вариант на обекта като например:

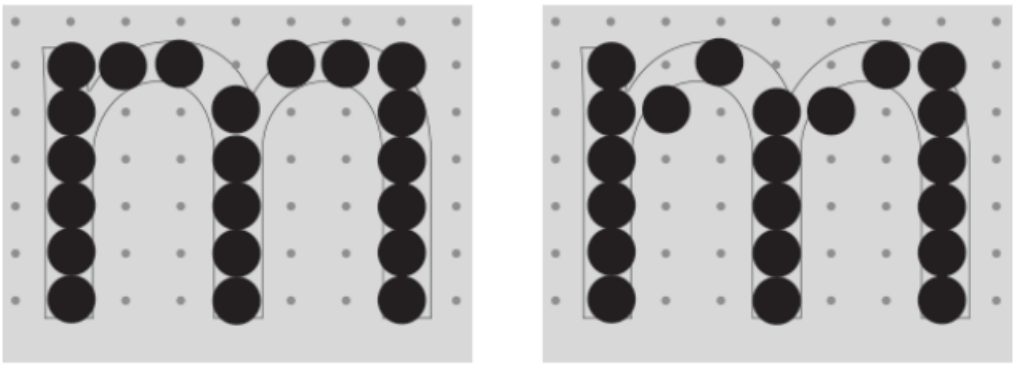


Фиг. 2

С развитието на технологиите специалистите са поставили филтри след пресмятането на позициите на пикселите, като резултата от това са по-чисти и хубави графики. За целта на този проект филтрите, които са по интересни са ***Hinting*** и ***Anti-Aliasing***.

- **Hinting[1]**

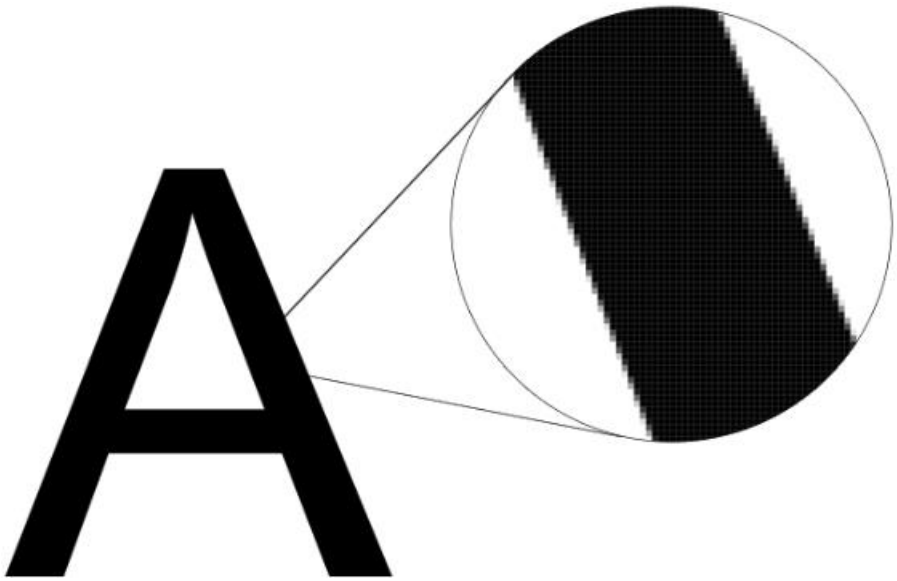
Както вече споменах с помощта на математически формули се изгражда „контура“ на желания символ. След това се изпълняват Hint инструкции, точно когато символа се изобразява на екрана. Тези инструкции разместват някой от точките, които определят символа, с цел да се подсигури правилно позициониране спрямо съответният грид на изобразяване. Това обаче по никакъв начин не променя визуалното възприемане на даден шрифт от човек, за нашите компютри обаче това са два съвсем различни обект[1].



Фиг. 3

- **Anti-Aliasing[1]**

Другият важен филтър за различаването на потребители е Anti-Aliasing[1]. Той е най-масовият филтър, който се използва днес. Неговата цел е да запълни ръбовете на символите със сиви пиксели за да изглеждат по-гладки. Това се дължи на начинът, по който човешките очи работят. При големи разлики в цветовата гама, нашите очи осредняват разликата в цветовете[1].



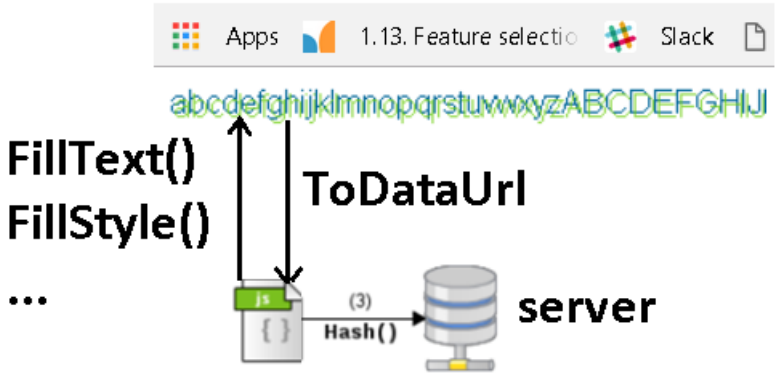
Фиг. 4

Когато на два компютъра се даде еднаква задача за рисуване, тези два филтъра се прилагат по различен начин в зависимост от посочените в **(3.1)** параметри. Като най-важни са хардуерните и софтуерните. Въпреки, че изглеждат еднакви резултатите от рисуването са напълно различими за компютъра. Колко уникални са обаче те ? Напълно неразличими ? В резултат на множество проучвания и установено, че в отделни случай машини с еднакви GPU могат да възпроизведат еднакви резултати. Каква точно е ентропията в такъв случай ? Това обаче ще се разгледам малко по-късно.[1]

3.3 Примерен fingerprint

```
<!DOCTYPE HTML>
<html>
   <head>
   </head>
   <body>
      <div id='yooo'></div>
      <img id="imageID">
         <canvas id="myCanvas" style="width:400;height:400;">
         </canvas>
      <script>
        window.addEventListener('load',fingerprint_canvas);
            function fingerprint_canvas() {
              "use strict";
              let  canvas, strCText, strText, strOut;
              canvas = null;
              strCText = null;
              strText =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
              strOut = null;
              canvas = document.createElement('canvas');
              strCText = canvas.getContext('2d');
              strCText.textBaseline = "top";
              strCText.font = "16px 'Arial'";
              strCText.textBaseline = "alphabetic";
              strCText.fillStyle = "#f60";
              strCText.fillStyle = "#069";
              strCText.fillText(strText, 2, 15);
              strCText.fillStyle = "rgba(102, 204, 0, 0.7)";
              strCText.fillText(strText, 4, 17);
              strOut = canvas.toDataURL();
              document.getElementById('imageID').src = strOut;
            }
      </script>
   </body>
</html>
```

Резултат от изпълнението на кода:



Фиг. 5

4 Как уефсайтовете получават тази информация? [1]

Уебсайтовете, които използват <canvas> дават инструкции на посетителите си да нарисуват определена фигура. Те обаче трябва да получат резултатът като данни, които могат да обработят. Директното изпращане на рендърнатата фигура би било непрактично, тъй като това струва скъпо. Решението на този проблем са hash функциите. Те дават възможността да се намали размера на изпращаната информация като запазват уникалността ѝ. Именно такава е идеята и на canvas отпечатъците – да изглежда напълно еднакво за човешкото око, но в самата структура да е напълно различно. Поради това hash функциите са перфектното решение.

Примерен код за изпращане на хешираната информация до сървър:

```
let md = forge.md.sha256.create();
                md.update(strOut);
                console.log(md.digest().toHex());
                let hashed_img = md.digest().toHex();
                $.ajax({
                  type: "POST",
                  url: "http:/localhost/Save_To_Server.php",
                  data: {
                     hashed_img: hashed_img
                  }
                   }).done(function(response) {
                       console.log('saved: ' + response);
                   });
```

```
<?php
    define('UPLOAD_DIR', '/fingerprints/');
    $img = $_POST['hashed_img'];
    $img = str_replace('', '', $img);
    $img = str_replace(' ', '+', $img);
    $data = base64_decode($img);
    $file = UPLOAD_DIR . uniqid() . '.png';
    $success = file_put_contents($file, $data);
    print $success ? $file : 'Unable to save the file.';
?>
```

5 Каква е ентропията ? [2], [3]

Сам по себе си Canvas Fingerprinting има 10 bits ентропия, това е ниско ниво на ентропия – от 1024 човека ще има 2 с идентични отпечатъка. Истинската сила на тази техника обаче идва от комбинацията и с други подобни техники. Например добавянето на характеристики на браузъра.

Attribute	Entropy(bits)
User agent	10.0
Accept Header	6.09
Plugins	15.4
Fonts	13.9
Screen resolution	4.83
Timezone	3.04
DOM storage	2.12
Cookies	0.353

На изображението в ляво е показана ентропията на различните характеристики, чрез които се формира браузър отпечатъкът. Средното ниво измерено за този тип отпечатък е 16 bits. Като съберем двете ентропии

заедно получаваме 26 bits ентропия – на всеки 67 млн. отпечатъка има 2 идентични. Това е доста добро ниво на ентропия.

https://browserleaks.com/canvas - тук може да се тества само canvas fingerprinting ентропията.

https://firstpartysimulator.org/tracker?aat=1&t=11&dnt=11 - тук всеки може да тества цялостната ентропията на своя браузър.

6 Как да се защитим от canvas fingerprinting? [2], [3], [4]

6.1 Блокиращи инструменти

Инструменти като Ghostery, No-Script и Privacy Badger могат да блокират скриптове за отпечатъци. По конкретно No-Script позволява да се изпълняват единствено скриптове позволени от потребителя. Ghostery и Privacy badger блокират свалянето на скриптове от сайтове, за които има информация, че правят подобни отпечатъци. Това обаче не е много ефективно решение тъй като е трудно да се поддържа актуална база данни. Допълнително, присъствието на подобни инструменти може да бъде засечено и това дава още информация, за по-лесно разпознаване на уникалността.

6.2 Използване на няколко браузъра

Друг вид решение е да се използват различни браузъри като по този начин всеки има различен отпечатък. Това обаче не е достатъчно понеже изграждането на отпечатъци се състой от данни за операционната система и хардуера, а те не се променят при смяната на браузъри.

6.3 Tor браузър

Идеята зад Tor браузърът е анонимност. Той опитва да създаде една универсална конфигурация за всеки потребител, за да не могат те да бъдат различени. Допълнително, когато се достъпва определен уебсайт Tor браузърът прехвърля потребителят през серия виртуални „тунели“ преди да го свърже с уебсайта и скрива IP адреса му. Най-значителната характерискита на този браузър обаче е това, че не спира всякакъв Javascript код и plug-ins, които могат да доведат до fingerprinting.

7 Заключение

Canvas отпечатъците са много силен инструмент за проследяване на интернет потребителя и защитата срещу него не е ефективна. За да бъде блокиран като инструмент за проследяване, това решение трябва да бъде прието от голяма група от хора, но това е малко вероятно, тъй като тази технология е много разпространена. Единственото по сигурно решение е Tor браузърът, но дори и той не е непробиваем, защото canvas използва рендър на шрифтове, а това не се предотвратява от Tor.

8 Източници

[1] https://multiloginapp.com/everything-need-know-canvas-fingerprinting/

[2] http://diversify-project.eu/papers/laperdrix15.pdf

[3] Pixel Perfect: Fingerprinting Canvas in HTML5, by Keaton Mowery and Hovav Shacham, Department of Computer Science and Engineering, University of California, San Diego, La Jolla, California, USA

[4] https://multiloginapp.com/two-approaches-combat-browser-fingerprinting/