Browser Fingerprinting via JavaScript DOM Mouse Events

Oliver Miller
Dept. of Computer Science
Purdue University
olivermiller@purdue.edu

Thomas Ibrahim
Dept. of Computer Science
Purdue University
ibrahimt@purdue.edu

***Abstract*—Our paper focuses on fingerprinting: the process of collecting data points to identify a user across multiple sessions and scenarios. More specifically, we analyze three browser generated events as our data points to identify certain device and OS specific behavior. We also adjust certain device and OS specific variables to identify whether these can be detected. By developing and application to collect these browser generated events, we were able to tell whether a user is 1) using a track pad or mouse and 2) using a low or high DPI mouse. This information could be applied in online advertising scenarios to help drive conversions (and thereby revenue) by more accurately targeting customers, and could be used in larger fingerprinting frameworks deployed across the web.**

I. Introduction

As the internet continues to play an increasing role in our lives, the need for robust security measures has grown in parallel to address the risks associated with these activities. In addition, the collection of data on the internet is more important than ever. The more data that business has, the more accurately they can target a user with advertisements, which means higher revenue for the company. One of the ways that a business can collect data is fingerprinting. Fingerprinting involves collecting unique data points from a user's device, to create a digital fingerprint that can be used to identify the user across multiple sessions and/or devices[1]. The diversity of the data that can be collected from a user's device has led to the development of numerous types of fingerprinting techniques. One example of fingerprinting is canvas fingerprinting. This technique uses the HTML5 canvas element to identify differences in a user's GPU, graphics drivers, or graphics card. This technique works by first letting the script write an image on a user's device. The browser then captured how the image was rendered by a specific user's device. Since computers have different hardware and drivers, the data collected can be slightly different. This data is then used to fingerprint users. Another technique used in fingerprinting is media device fingerprinting. This technique explores the idea of keeping a list of all the connected media devices and their respective IDs on a user's laptop or PC. This list is then analyzed to see if a user can be fingerprinted by their connected devices. While new and innovative fingerprinting techniques continue to develop, the specific method of fingerprinting that this paper will cover will be fingerprinting using specific mouse movement DOM events. In particular, onMouseMove, onWheel, and onScroll events will be examined to determine if they are useful as a fingerprinting identifier for users' behavior.

II. Background

To fully understand the succeeding procedure, some key concepts must first be explained. We will initially explain the idea of DOM and DOM events. Next, the specific DOM events used will be discussed. The device settings: DPI and scroll bar, will be explained, as these act as our different device configurations.

A. DOM

The document object model is a data representation of all the objects that make up an HTML website. The DOM represents the documents as many different nodes and objects. These nodes and objects are rendered on the screen by the browser, and JavaScript can be used to make these elements more interactive. The programming language we used for this paper is JavaScript.

B. DOM Events

DOM events enable JavaScript to add event handlers to HTML elements. These DOM events allow us to track when a change that we are tracking is triggered[3]. We examined three specific DOM events:

- MouseMove: The MouseMove event is fired at an element when a mouse is moved. The two specific MouseMove properties that were analyzed are MouseEvent.movementX and MouseEvent.movementY. MouseEvent.movementX is the change in the x coordinate relative to the position of the last MouseMove event. MouseEvent.movementY is the change in the y coordinate relative to the position of the last MouseMove event.
- Wheel: The wheel event fires when the user rotates a wheel button on a pointing device. The specific wheel property that was analyzed was deltaY. DeltaY is a double that represents the vertical scroll amount in between wheel events.
- Scroll: The scroll event fires when an element has been scrolled prior to the last event. The specific property that was analyzed is deltaT. The deltaT is the change in time between each scroll event.

C. DPI & Scroll Bar

Dots per inch or DPI is the measurement of the speed that a mouse moves on screen in relation to how much the user actually moved his mouse. A low DPI means low mouse speed and a high DPI means high mouse speed. [4] In this paper, DPI is analyzed to see if a different DPI's will lead to data that can lead to the fingerprinting of users. The scroll bar is a setting that can be changed on computers. This setting dictates how much vertical distance is changed when a user scrolls. In this paper, different levels of the scroll bar are analyzed to see if the data can be used to fingerprint users.

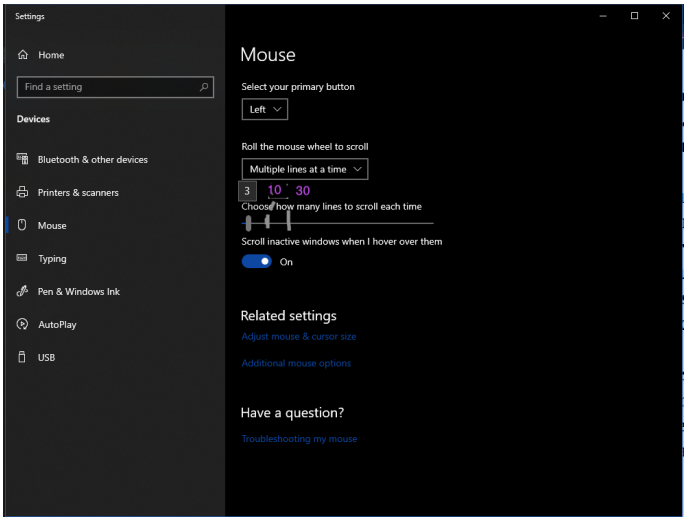

Fig. 1. Scroll Settings

Figure 1 shows the different scroll settings that were adjusted to collect data. The regular setting was three lines per scroll. The high setting was 10 lines per scroll. The extreme setting was 30 lines per scroll.

III. Procedure & Implementation

To collect the necessary statistics from different device configurations, a testing webpage was created. A variety of device configurations were tested on the webpage, where the relevant statistics were saved for the analysis portion of the project.

A. Developing the Testing Interface

After deciding which DOM events to target, and how best to utilize certain HTML elements to trigger the events, developing the scenario webpages was straightforward. Three skeleton (HTML) pages exist in addition to three accompanying JavaScript files that correspond on a one-to-one relationship for each HTML skeleton page.

1) playgroundA.html ⇔ playgroundA.js
   Playground A targets the "MouseMove" event on the "HTML" DOM element. Any mouse cursor movement within the bounds of the page will log the DOM event.
2) playgroundB.html ⇔ playgroundB.js
   Playground B targets the "Wheel" event on the "HTML" DOM element. Any scrollwheel movement within the bounds of the page will log the DOM event. Additionally, if already at the top or bottom of the page, scrolling will be disabled by the browser, therefore we created several pagelengths of Lorem Ipsum content to provide the user with plenty of lengths to scroll.
3) playgroundC.html ⇔ playgroundC.js
   Playground C targets the "Scroll" event on "H1" DOM element. Any scroll movement within the bounds of the scrollable "H1" element will log the DOM event.

B. Deploying the Testing Interface

Our source code existed on GitHub, but we wanted our testing interface to be easily accessible across devices to ensure that trying multiple device configurations was frictionless. We used Vercel.com to host our webpages, as it automatically receives changes from our GitHub repository and deploys them to the web without additional configuration. Another convenience of Vercel is that our repository is assigned a subdomain, ours is publicly accessible at mouse-fingerprinting.vercel.app.

C. Testing Scenario Procedure

1) Playground A:

- Testing Device sets environment to desired specifications. This includes changing the DPI to the desired amount along with the scroll setting.
- Testing Device accesses Playground A
- Testing Device pointer starts out-of-frame (off the page) and in one swift motion moves the pointer to the button and presses it.
- Data is now copied to the Testing Device's clipboard in CSV (comma-separated values) format. The copied data is now ready to be analyzed in a supported application (e.g. Excel).

2) Playground B:

- Testing Device sets environment to desired specifications (DPI/scroll setting).
- Testing Device accesses Playground B
- Testing Device pointer starts on the page. In one or several swift motions, the device scrolls to the button and presses it.
- Data, similar to above, is ready for extraction and analysis.

3) Playground C:

- Testing Device sets environment to desired specifications (DPI/scroll setting).
- Testing Device accesses Playground C
- Testing Device pointer starts inside the bounds of the paragraph and in one swift motion scrolls to the end. The button is pressed.
- Data, similar to above, is ready for extraction and analysis.

D. Data Extraction

After performing the *Testing Scenario Procedures*, the CSV data was pasted into an Excel document for each testing configuration. After all the data was compiled for each testing scenario and device configuration, it was organized on a directory basis. Each primary level folder contained the device name, and each secondary level folder contained the device configuration.

```
mouse-fingerprinting/
|-- index.html
|-- playground(A|B|C).html
|-- playground(A|B|C).js
`-- data_collection/
    |-- logitech_g305/
    |   |-- 800_dpi_extreme_scroll/
    |   |   |-- a.csv
    |   |   |-- b.csv
    |   |   `-- c.csv
    |   |-- 800_dpi_high_scroll*
    |   |-- 800_dpi_regular_scroll*
    |   |-- 1600_dpi_regular_scroll*
    |   `-- 3200_dpi_regular_scroll*
    |-- logitech_g900*
    `-- pixelbook_trackpad*
```

Note: The asterisk indicates a similar directory structure as the preceding instance; it has been omitted for brevity.

IV. Results

A. Wheel Event

The first set of attributes that we decided to analyze was the difference in the wheel event on a track pad versus a mouse. First we graphed two 800 dpi track pads (MacBook and Pixelbook) and two 800 DPI mice (Logitech G900 and Logitech G305)

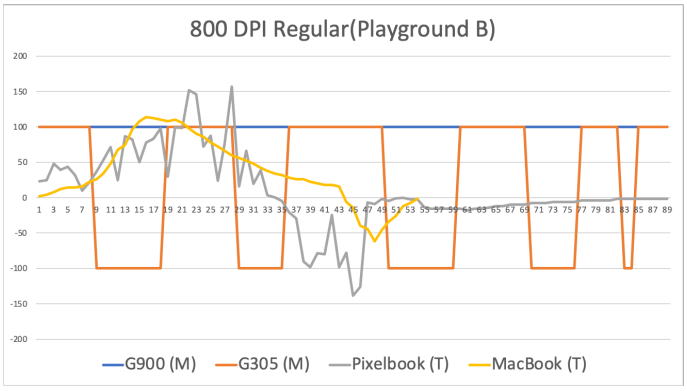

Fig. 2. Track Pad vs Mouse

From figure 2, it can be observed that both mice move in block-like movement. While on the other hand, the track pads' movements are much more constant and less spaced apart. Next we decided to look at the same attributes but with a 3200 DPI setting to see what changes that would cause.

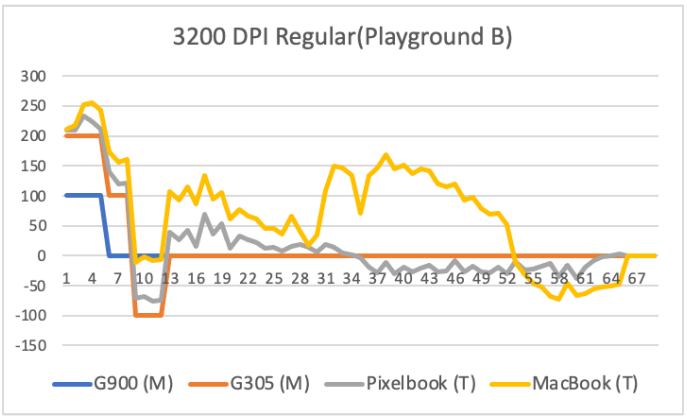

Fig. 3. Track Pad vs Mouse (3200 DPI)

We noticed that changing the DPI did not affect the graph significantly and the track pad and mice shape stayed the same.

B. MouseMove Event

The next event we analyzed was the mouse move event. Specifically, we were interested to see if different DPI changed the value of DeltaX. Below is a graph of our results. Both the Logitech G305 mouse and Logitech G900 mouse were analyzed at 800 DPI and 3200 DPI.

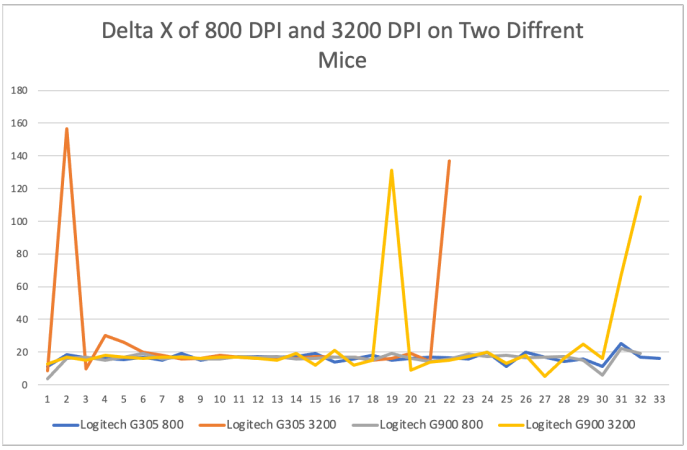

Fig. 4. Low vs High DPI

As seen in figure 4, the only time that deltaX would exceed 20 was with a 3200 DPI mice. In addition, the only time deltaX would have such an extreme change was with a 3200 DPI mouse.

C. Scroll Event

Thinking that mice and track pads would update at different speeds, we analyzed the scroll event. Specifically, we analyzed and graphed the time differential between the current and previous event. We observe in Figure 5 (next page) the two mice have an incredibly similar update speeds, as they even overlap each other. This contrasts with the track pads which are significantly "pointier" and are a lot less consistent.

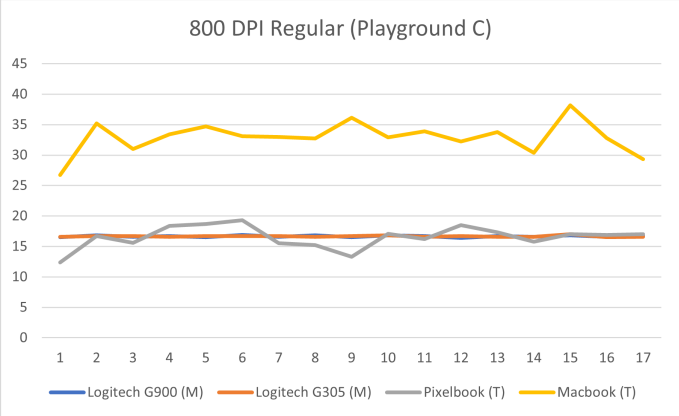

Fig. 5. Scroll timing delta of Mouse vs. Track pad

V. Discussion

Targeted advertising has been proven to lead to greater conversions, resulting in 40% higher probability of a successful financial conversion [2]. We discuss two possible scenarios of how our results could tangibly be used to target certain types of users, and how advertisers could exploit these techniques.

A. Targeting Laptop vs. Desktop users

The wheel event can be used to learn whether a user is on a track pad or a mouse. This can be seen in Figure 2 and Figure 5. This information can be used to target better advertisements to users. If a user is identified as a track pad user, it makes logical sense to advertise them with laptop exclusive products. Take for example an advertisement about a laptop sleeve, laptop charger, or laptop stickers. This is beneficial to a business because more relevant advertisements mean better conversions – resulting in more revenue. In addition, the wheel event can be used as a proxy to identify whether a user is at a workplace. At many workplaces, a mouse is used instead of a track pad, and this metric could be valuable in addition to others to increase the confidence to advertisers whether a user is at work or not. This information can be used to target better advertisements to users at work.

B. Targeting Regular Users vs. Gaming Enthusiasts

Figure 4 leads us to the realization that we can discriminate whether a user is using a higher DPI. This information is most relevant in the gaming world, where gaming enthusiasts like to use a high DPI to improve gaming performance. This information can be used to target more gaming advertisements to users who are most likely to buy the product. Inversely, users who are using a very low DPI should not be getting gaming advertisements because they are less likely to game frequently.

VI. Conclusion

A. Our Progress

Mouse related fingerprinting can be used to identify a user, which in of itself is not particularly revolutionary–many techniques are already employed that allow websites and adversaries to fingerprint a user across multiple sessions. Despite this, providing additional metrics that we have researched can help increase the robustness, reliability, and permanency of an already in use fingerprinting framework. In other words, our metrics, data, and analyses are not impenetrable on their own, but could be combined with other metrics to make an incredibly strong system to track users. Even just using our metrics by themselves, there are certain valuable insights for advertisers and would-be fingerprinters. For example, we determine whether the user uses a high or low DPI mouse– a relevant statistic in the gaming industry for advertisers. Furthermore, we were able to determine likely indicators on whether the user is using a mouse or a track pad.

B. Outlook

In the future, we anticipate that the use of fingerprinting will grow as part of targeted advertising. It has already been proven that target advertisements result in more revenue for a business, therefore we can only expect that it is only a matter of time until fingerprinting becomes more ubiquitous for targeted advertising. We also anticipate more security features will be implemented in web browsers soon, in order to combat the increasing use and development of fingerprinting technologies. Users do not like to be tracked, and will be a demand for a more secure and privacy conscious browser. We anticipate increasingly advanced techniques from both sides to be deployed.

VII. Further Reading

While our scope was relatively limited, there are many additional ways (some mentioned earlier) that allows fingerprinting to be performed via other methods. We provide some additional methods on how fingerprinting could be used with other techniques, yet are still within the scope of the browser.

- Canvas Fingerprinting
- Canvas Fingerprinting in Web Assembly
- Audio Fingerprinting
- WebGL Fingerprinting

References

[1] Savannah Copland. The top browser fingerprinting techniques explained - fingerprint, Apr 2021.
[2] Ayman Farahat and Michael C. Bailey. How effective is targeted advertising? In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, page 111–120, New York, NY, USA, 2012. Association for Computing Machinery.
[3] MozDevNet. Introduction to the dom - web apis: Mdn, Feb 2023.
[4] Leo Parrill. What is dpi, and why is it important?, Dec 2022.