CPTS 580 HW6
Peek-a-Boo, I See You

1 Overview

Welcome to the sixth assignment for Web Security. In this assignment, you will develop your fingerprinting techniques for web tracking.

2 Preparation

2.1 check Node.js

You should already have Node.js installed from previous assignments. Please confirm that Node.js is installed. If not, you can install Node.js from the official website.

2.2 Get starter code

Download the starter code file 'hw6-main.zip' from Canvas. Extract the files and enter the folder. Install the necessary dependencies with npm:

```
node install
```

Start the server:

```
node start
```

Your browser should open up to http://localhost:4001/, where you can begin the assignment. The page should display the fingerprint 8854-4c55-3047-2bd4 in the middle, corresponding to a hash of the placeholder string 'identifier' in the starter code.

3 Coding Portion

3.1 Instructions

For this assignment, the only coding file you need to modify is 'fingerprint.js'. Your goal is to implement fingerprinting techniques in 'fingerprint.js' that generate a unique identifier for a user across multiple browsing sessions.

The return value of your fingerprint function should be a string, likely a hash of various fingerprinting vectors. A hash function, defined in 'hash.js', is provided for your use. This function can accept any number of arguments, each of which can be a string, number, JSON object, etc., and returns a hash corresponding to those inputs. You can check the return value of your fingerprint function by visiting http://localhost:4001/ in your browser. Additionally, feel free to use *console.log* to help with debugging; these can be viewed in the browser's Developer Tools console. Note that fingerprinting should be done without requiring user interaction.

You may also utilize CSS on the client side, leverage server-side code, or deploy anything else you think might be helpful for your fingerprinting. This is optional and not necessary to receive full credits.

3.2 Requirements

You must use at least **5** distinct fingerprinting vectors. You may NOT use the client's User-Agent as one of your fingerprinting vectors. (In practice, it's often used, but makes this exercise too simple.)

3.3 Grading

We will grade your fingerprinting techniques under the following situations and award points if the specified criteria for the return value of 'fingerprint.js' are met.

1. Returns the same identifier (in Google Chrome) when the provided HTML page is opened, the tab is closed, a new tab is opened, and the page is visited again. (15%)
2. Returns a different identifier when browsing to the page from a different browser (Firefox or Safari), serving as a proxy for different users. (15%)
3. Returns the same identifier in Chrome, even after browser data (e.g., cookies, cache, localStorage, etc.) is cleared in Google Chrome's settings. (15%)
4. Returns the same identifier even after clearing browser data in one additional browser (Firefox or Safari). (15%)
5. Returns a different identifier when accessing the page from Chrome on your device versus Chrome on another device. This ensures your fingerprinting techniques distinguish between individual devices, not just browser types. (10%)
6. Returns the same identifier across sessions in incognito/private browsing mode. (5%)
7. Implement any new fingerprinting vector not listed in Section 5.(Extra credit: 10%)

4 Short Answer Questions

Your answers should be concise. Include your answers in a text file (doc, pdf, txt). (5% per question)

1. What fingerprinting methods did you use? Why did you choose them?
2. What are the current limitations of your fingerprinting implementation, and under what circumstances would it fail?
3. Suppose you are developing a privacy-preserving web browser. Describe ONE strategy to defend against your fingerprinting, or explain why your method is unable to be defended against.
4. Analyze the costs of your proposed mitigation strategy, in terms of performance, user experience, and web compatibility.
5. The following privacy-preserving browsers have implemented anti-fingerprinting protections. Test your fingerprinting techniques with **one** of these browsers. Does it defend against your fingerprinting techniques? If so, how could you modify your techniques to remain effective despite these countermeasures?
   (a) Brave Browser (default settings)
   (b) Firefox Browser (Enable Fingerprinting protection)
   (c) Tor Browser (default settings)

4.1 Submission

Please upload the text file, 'fingerprint.js', and any other files you've used for fingerprinting, if applicable. For instance, if you've created a new client-side file or modified the server code in 'your-server-code.js', ensure you submit that file as well.

5 Resources

1. AmIUnique
2. fingerprintjs