Web Browser Fingerprinting: Revealing Stealth Tracking Techniques and Defenses

Xu Lin
Washington State University
CySER May 22, 2024
What is Web Tracking?
Why Does Tracking Exist?
Classic Tracking

Cookies

- Website stores an id on the client
- The client returns the id to the server
- The id is what allows re-identification
- “Stateful”
Do websites still know who you are?
What Is Browser Fingerprinting

“A device fingerprint, machine fingerprint, or browser fingerprint is information collected about a remote computing device for the purpose of identification. Fingerprints can be used to fully or partially identify individual users or devices even when cookies are turned off.”

- Browser fingerprints can be trivially collected by any website the user visits through a series of JavaScript APIs.
Visit page

Send fingerprints

Page with fingerprinting script

Identify user

CySER
Browser fingerprinting is a double-edged sword

Usage scenarios

● Security Enhancement
  ○ Bot Detection: Differentiate between bots and legitimate web users
  ○ Authentication: Differentiate between a legitimate owner of an account and someone impersonating that owner

● Privacy Concerns
  ○ Track users against their will (stateless tracking)
Risk-Based Authentication and Two-Factor Authentication (2FA)

- 2FA creates friction for users
- Certain websites only trigger 2FA for suspicious login attempts
Advanced Risk-Based Authentication That Uses Browser Fingerprinting

1. Visit login page
   - https://www.target-website.com

2. Page with fingerprinting script
   - Successful Login
   - OK

3. Send login, password, fingerprints
   - Fingerprints match
   - Fingerprints don't match

4. Grant access or trigger 2FA
   - Check Your Mobile
   - A code has been sent to (123)456-7890.
   - Verify
   - Successful Login
   - OK

Check Your Mobile
- A code has been sent to (123)456-7890.
- Verify
- Successful Login
- OK

Fingerprints don't match
- Fingerprints match

https://www.target-website.com

Advanced Risk-Based Authentication That Uses Browser Fingerprinting
Legitimate Login

Bypass 2FA?

https://www.target-website.com
Overview of our attack workflow

Figure 1: Overview of our attack workflow that misuses browser fingerprints for bypassing ancillary security checks.
Phase 1: attacker visits target websites and "extracts" their fingerprinting code

1. Enable FP-extractor extension
2. Visit target-website
3. "Extract" fingerprinting code
4. Automatically replicate the exact fingerprinting process of target websites

Authentication Pitfalls of Browser Fingerprinting
Phase 2: attacker obtains user’s credentials and fingerprints

1. Deploy phishing site
2. Visit phishing website
3. Page generates fingerprints of user's device
4. Collect login, password and fingerprints

Generate fingerprints identically to the ones expected by target websites
Phase 3: attacker spoofs fingerprints and bypasses 2FA mechanism

1. Enable FP-Spoofer extension
2. Visit target website
3. Page with fingerprinting script
4. Spoof fingerprints, send login, password
5. Grant access or trigger 2FA

Fingerprints Match !!!
What about the real world?
## Risk-based authentication mechanisms in popular web services

<table>
<thead>
<tr>
<th>Website</th>
<th>BasicFP</th>
<th>Fingerprinting Technique</th>
<th>Audio</th>
<th>IP Address Restrictions</th>
<th>Vulnerable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Canvas/WebGL</td>
<td>Fonts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank-A</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Bank-B</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>-</td>
</tr>
<tr>
<td>CreditCard</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Trading-A</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Trading-B</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Tax-A</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Tax-B</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Tax-C</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Tax-D</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>eCommerce-A</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>eCommerce-B</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>RideSharing</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Food&amp;Beverage-A</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Food&amp;Beverage-B</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
</tbody>
</table>

- **We completely bypass** 2FA in 9/14 websites that use FPs for authentication!
- Attack only prevented by IP address checks.
- We inject X-Forwarded-For header (used by proxies) with the user’s IP to bypass IP-checks (→).
- Certain sites only require an IP from the same city (○).
Browser fingerprinting is a double-edged sword

Usage scenarios

● Security Enhancement
  ○ Bot Detection: Differentiate between bots and legitimate web users
  ○ Authentication: Differentiate between a legitimate owner of an account and someone impersonating that owner

● Privacy Concerns
  ○ Track users against their will (stateless tracking)
Online tracking

Browser fingerprinting heavily relies on JavaScript.
Fingerprinting countermeasures

Privacy-focused browsers and anti-fingerprinting extensions
- Spoof certain APIs
- Disable JavaScript (entirely or partially)
Is fingerprinting possible **without** JavaScript?
Our approach: Implicit stylistic browser fingerprinting

- Does not use any JavaScript
- Provides highly discriminating fingerprints
What can we use to detect the stylistic differences?

<table>
<thead>
<tr>
<th>Chrome</th>
<th>Firefox</th>
<th>Safari</th>
</tr>
</thead>
<tbody>
<tr>
<td>40px/15px</td>
<td>183px/16px</td>
<td>34px/13px</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>12-hour Time</th>
<th>24-hour Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>146px/32px</td>
<td>99px/32px</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>English OS</th>
<th>Chinese OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>425px/35px</td>
<td>425px/41px</td>
</tr>
</tbody>
</table>
Fingerprinting attributes

Certain HTML elements have different sizes depending on certain environmental factors.

<table>
<thead>
<tr>
<th>Category</th>
<th>Fingerprint attributes</th>
<th>AIU</th>
<th>FPJS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>browser</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>browser major version</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>operating system</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>platform</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td>operating system language</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>scrollbar settings</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JS disabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fonts</td>
<td>font preferences</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td></td>
<td>supported fonts</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>supported shadow fonts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ad blocker</td>
<td>presence of ad blocker</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ad blocker identification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Media properties</td>
<td>screen resolution</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>supported media features</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td></td>
<td>media features’ values</td>
<td>○</td>
<td></td>
</tr>
</tbody>
</table>

AIU: captured by AmlUnique  FPJS: captured by FingerprintJS
●: full feature support  ○: partial feature support
• The page only needs **25** iframes.
• All elements are placed in an 800px by 1000px iframe (main iframe) to ensure that their dimensions remain consistent across different browser window sizes.
Other Tracking Vectors: Using Extensions

Carnus: Exploring the Privacy Threats of Browser Extension Fingerprinting

Soroush Karimi, Panagiotis Illa, Konstantinos Solomos, Jason Polakis
University of Illinois at Chicago, USA
(skarim5, pilla, ksolom, polakis)@uic.edu

Abstract—With users becoming increasingly privacy-aware and browser vendors incorporating anti-tracking mechanisms, browser fingerprinting has garnered significant attention. Accordingly, prior work has proposed techniques for identifying browser extensions and using them as part of a device’s fingerprint. While previous studies have demonstrated how extensions can be detected through their web accessible resources, there exists a significant gap regarding techniques that indirectly detect extensions through behavioral artifacts. In fact, no prior study browsers still mediate a large portion of our online activities. As a result, the evolution of websites from static resources to functionality-rich applications has also necessitated the evolution of browsers into complex platforms with a rich set of APIs and features. To improve user experience, browsers allow users to further personalize them and extend their functionality by installing extensions.

Fingerprinting in Style:
Detecting Browser Extensions via Injected Style Sheets

Pierre Laperruixe
Univ. Lille, CNRS, Inria
Oleksiw Starov
Palo Alto Networks
Quan Chen
North Carolina State University
Alexandros Kapravelos
North Carolina State University
Nick Nikiforakis
Stony Brook University

Abstract
Browser extensions enhance the web experience and have seen great adoption from users in the past decade. At the same time, past research has shown that online trackers can use and a browser’s private mode) stateless tracking techniques anise that enable third parties to track users across sessions, without relying on previously set cookies or other stateful identifiers. These stateless techniques essentially “fingerprint”
Other Tracking Vectors: Using Favicons

Tales of FAVICONS and Caches: Persistent Tracking in Modern Browsers

Konstantinos Solomos, John Kristoff, Chris Kanich, Jason Polakis
University of Illinois at Chicago
{ksolomos, jkristoff, ckanich, jpolakis}@sic.edu

Abstract—The privacy threats of online tracking have garnered considerable attention in recent years from researchers and practitioners. This has resulted in users becoming more privacy-cautious and browsers gradually adopting countermeasures to mitigate certain forms of cookie-based and cookie-less tracking. Nonetheless, the complexity and feature-rich nature of modern browsers often lead to the deployment of seemingly innocuous functionality that can be readily abused by adversaries. In this paper we introduce a novel tracking mechanism that mimics a simple yet ubiquitous browser feature: favicons. In more detail, a website can track users across browsing sessions by storing a tracking identifier as a set of entries in the browser’s dedicated favicon cache, where each entry corresponds to a specific subdomain. In subsequent user visits the website can reconstruct the identifier by observing which favicons are requested by the browser while the user is automatically and rapidly redirected through a series of subdomains. More importantly, the caching of favicons in modern browsers exhibits several unique characteristics that render this tracking vector particularly powerful, as it is persistent (not affected by users clearing their browser data), non-destructive (reconstructing the identifier in subsequent visits does not alter the existing combination of cached entries), and even crosses the isolation of the incognito mode. We experimentally evaluate several aspects of our attack, and present a suite of

I. INTRODUCTION

Browsers lie at the heart of the web ecosystem, as they mediate and facilitate users’ access to the Internet. As the Web continues to expand and evolve, online services strive to offer a richer and smoother user experience; this necessitates appropriate support from web browsers, which continuously adopt and deploy new standards, APIs and features [76]. These mechanisms may allow web sites to access a plethora of device and system information [55], [21] that can enable privacy-invasive practices, e.g., trackers leveraging browser features to exfiltrate users’ Personally Identifiable Information (PII) [24]. Naturally, the increasing complexity and expanding set of features supported by browsers introduce new avenues for privacy-invasive or privacy-violating behavior, thus, exposing users to significant risks [53].

In more detail, while cookie-based tracking (e.g., through third-party cookies [57]) remains a major issue [29], [9], [69], tracking techniques that do not rely on HTTP cookies are on the rise [63], [16] and have attracted considerable attention from the research community (e.g., novel techniques for tracking and browser fingerprinting [72], [71], [13], [14], [12], [11]).
“On the Internet, nobody knows you’re a dog.”

“Remember when, on the Internet, nobody knew who you were?”
If you’re interested in breaking and fixing on the Web, please contact me at: xu.lin@wsu.edu