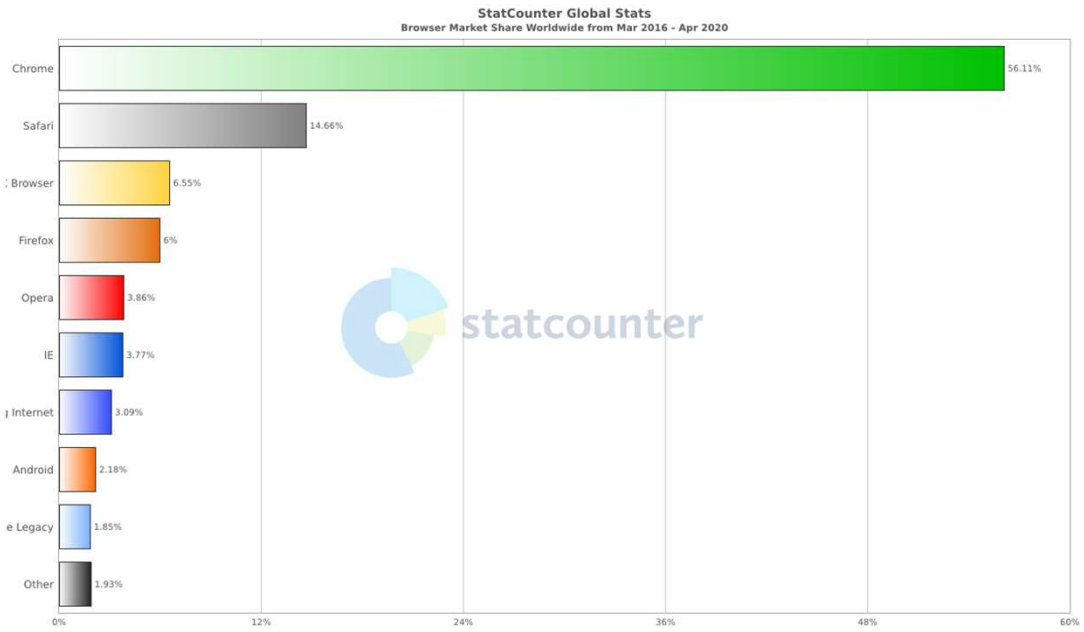
Browserprint: An Analysis of the Impact of Browser Features on Fingerprintability and Web Privacy

Seyed Ali Akhavani, Jordan Jueckstock, Junhua Su, Alexandros Kapravelos, Engin Kirda, Long Lu

Northeastern University, NC State University

ISC 2021

Browser Market Share

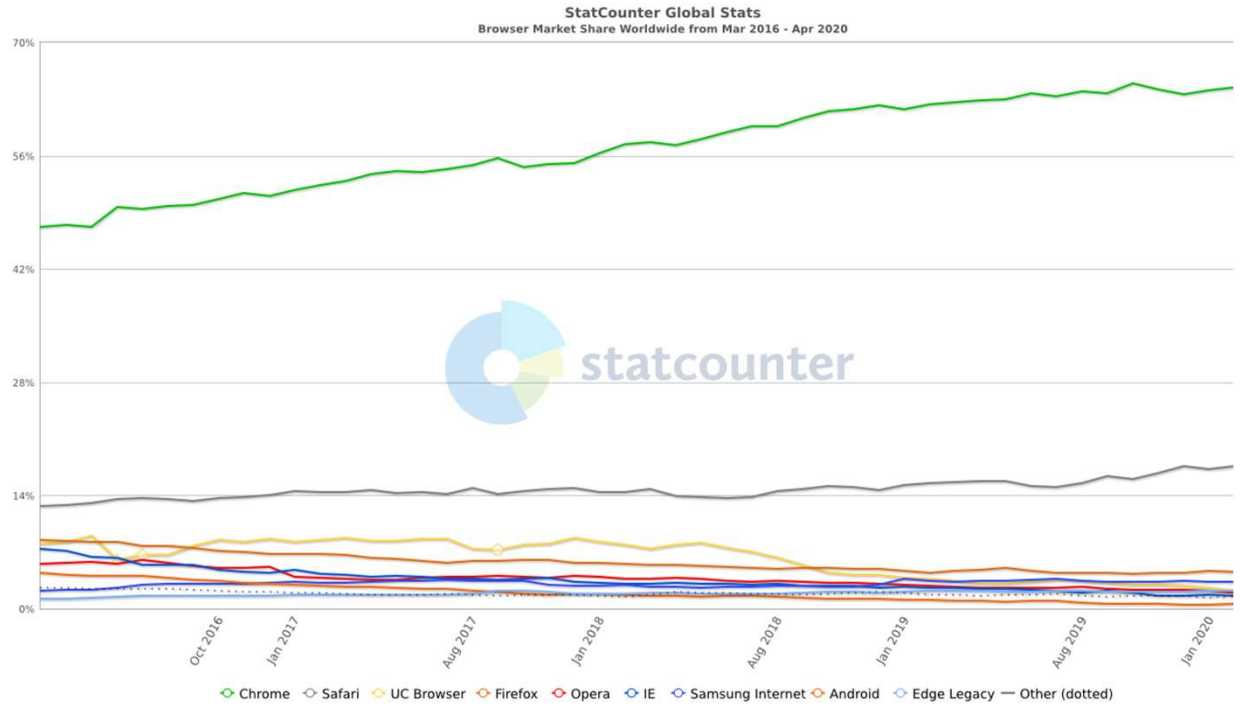


Terminology

- Browser Fingerprinting
  - Many unique details of a user's browser such as its hardware, operating system, browser configuration and preferences can be exposed through the browser
  - An attacker who collects and sums these outputs can create a unique **fingerprint** for tracking and identification purposes
- Browser features
  - All functionality that is available to attackers directly through JavaScript
  - These are the root problem of most web attacks
- Fingerprintability
  - Ratio of browser features that are associated with fingerprinting techniques in a browser version.

Introduction

- Chrome has the highest market share between browsers
- Safari, Firefox, and Opera are the next browsers with high market share
- Opera and Chrome are both Chromium-based and share the same codebase



Research Questions

- Are major versions of Firefox, Chrome, and Opera browsers fingerprintable based on their feature sets?
- Are these browsers becoming more fingerprintable over time?
- With respect to browser bloating, how does Firefox compare to Chrome and Opera?
- Could the private window mode reduce the possibility of being fingerprinted?
- Is there any noticeable difference between Chrome and Opera in case of fingerprintability?

Feature Gathering

- **Feature:** JavaScript objects, methods, and property values built into the global namespace of the browser's JavaScript implementation
- Crafted a JavaScript instrumented webpage that analyzes the visiting browser when it is visited.
- The code probes and iterates through the features supported by the browser
  - Using JavaScript to traverse the tree of non-cyclic JavaScript object references accessible from a pristine *window* object
  - Collects the full feature names
  - Each feature name comprises the sequence of property names leading from the global object to a given built-in JavaScript value.
- Captured feature sets are then stored in a database, tagged with identifying metadata

Testing Platform

- Used BrowserStack platform to visit the feature extractor webpage in headful mode
  - Cloud-based browser testing platform
- All of the tests are run on a single device with a single device configuration and OS
- Include all the major browser versions release during March 2016-April 2020
  - Chrome 49-81
  - Firefox 45-75
  - Opera 36-68

Browser Fingerprinting APIs

- We generate a list of suspicious APIs associated with fingerprinting
- Contains 313 JavaScript APIs
- These APIs provide functionality. However, they can be abused by creating a unique fingerprint of the client's browser.
- Literature Review
  - Extract suspicious APIs discussed in prior works: Panopticlick, AmIUnique, Hiding in the Crowd, and FPDetective.
  - These APIs Form 10% of the list
- Experimental Analysis

Experimental Analysis

- Crawling websites and Extracting Suspicious APIs from the data
- A crawler that visits EasyList domain file. Contains 13,241 domains
- Processed the raw data and collected all the API usages.
  - API usage of 8,682 domains with 56,828 origins was collected
- Manual Analysis to check if the APIs actually expose user information
  - Check Mozilla's MDN web docs
  - API is classified as a suspicious fingerprinting API if it can provide the information to filter certain users out
- Keyword search in all the crawled domains (BatteryManager)
- Expand the list by visiting known fingerprinting websites such as amiunique.org

Limitations

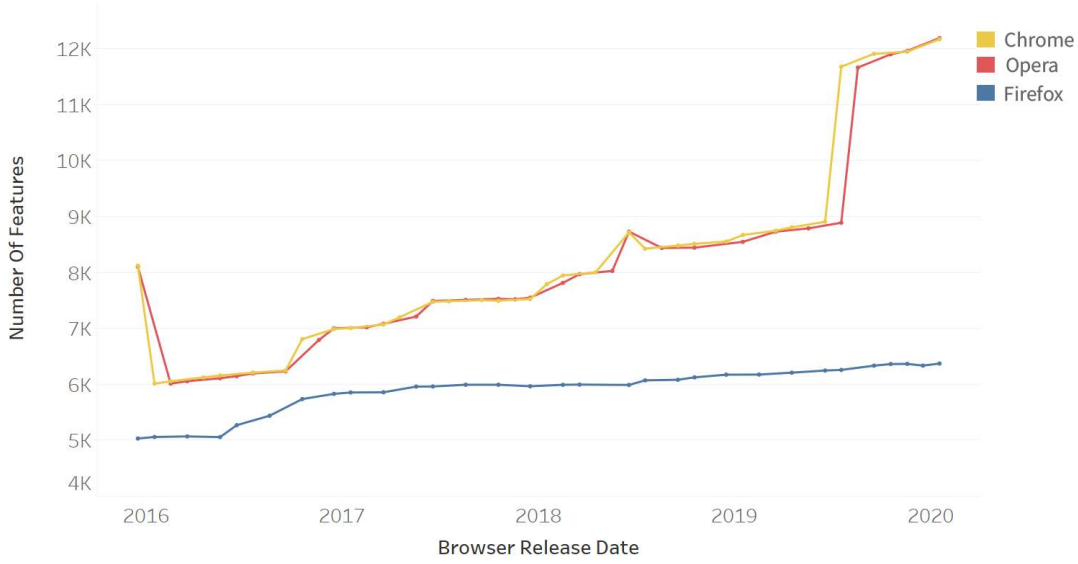
- Might not have all the fingerprinting APIs since we only crawled EasyList domains
- Manual Analysis might be affected by misonceptions between what is discussed in the Mozilla API page and the real API usage

Results

- Analysis of the Browser Features
- Browser Fingerprintability
- Unique Feature Set

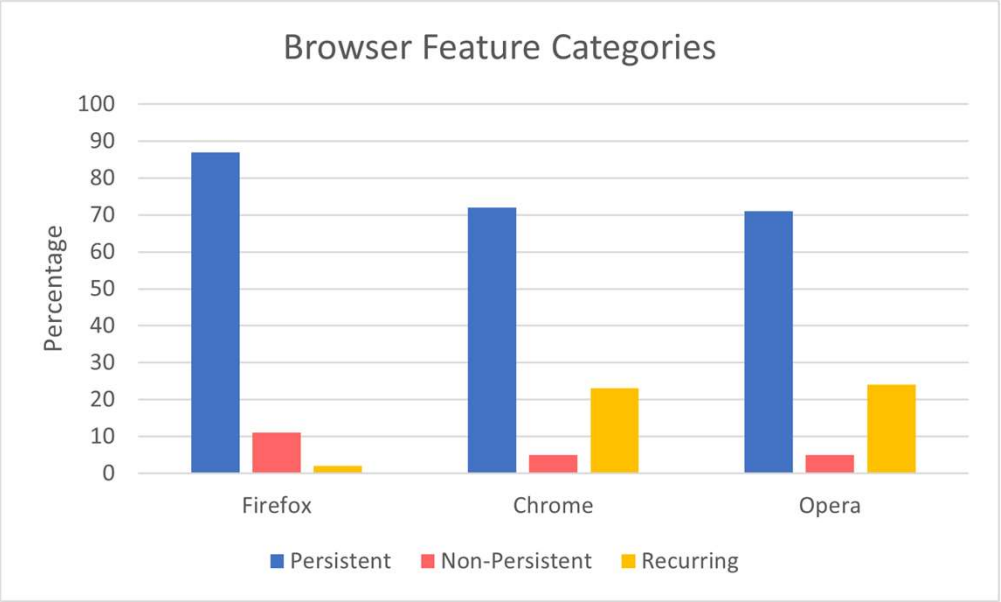
Feature Trends

- Browsers are becoming more bloated over time.
- Chrome and Opera share the same codebase. The trends are similar but minor differences exists between them
- Firefox contains much less features compared to Chromium-based browsers

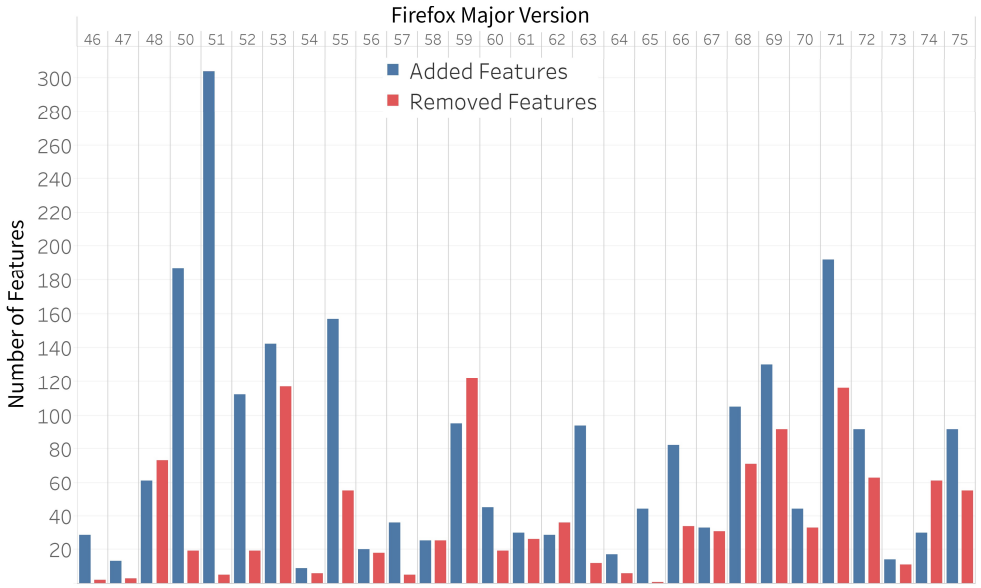


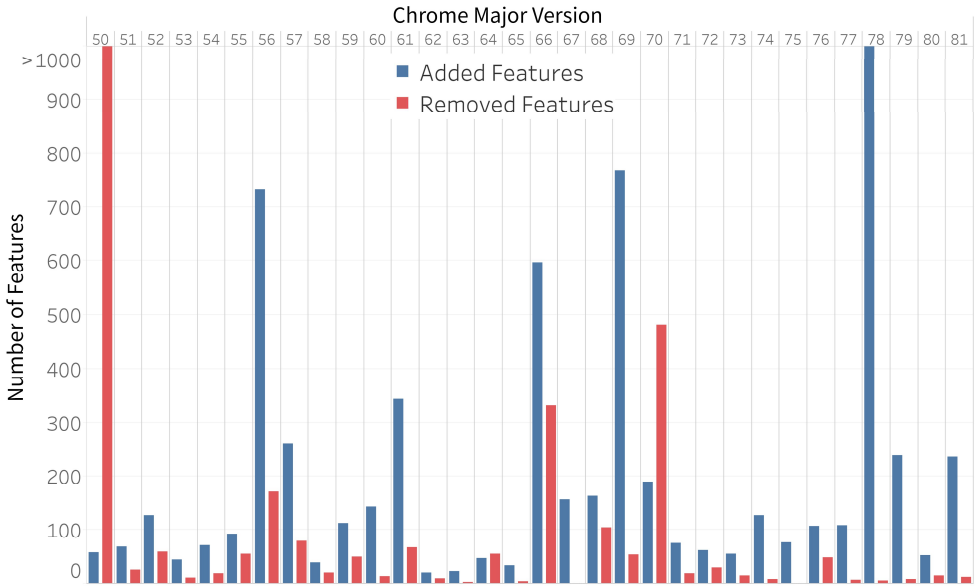
Feature Categories

- Persistent Features
  - Added to a specific version, and that continue to exist
- Non-Persistent Features
  - Existed in older versions of the browser, but were removed, and never appeared again
- Recurring Features
  - Added and removed from the browser from time to time



Feature Add and Removals

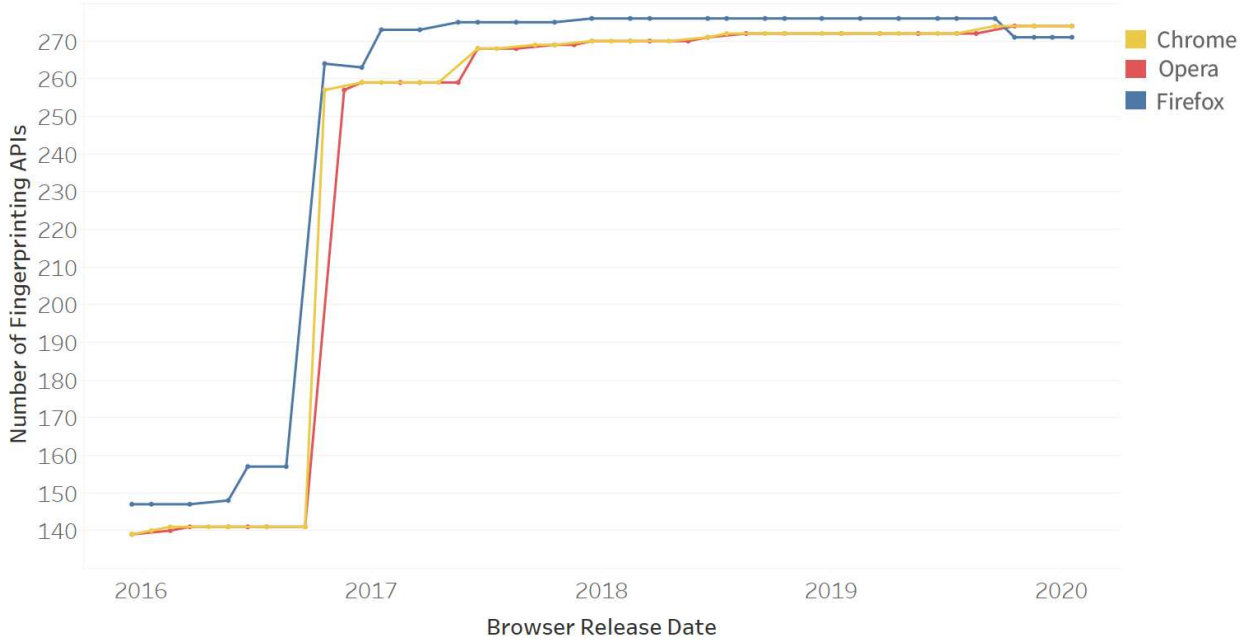




- Although both browsers are having more features over time, Adding and Removing Feature Trends are different.
- Firefox tries to maintain its total number of features

Browser Fingerprintability

- Browsers are becoming more fingerprintable over time
- Chrome 81 has 274 fingerprinting APIs! (The total number of suspicious APIs is 313)
- Adding more features is leading to having more fingerprintable APIs



Analyzing Spikes in the Graph

- January 2017
  - *WebGL 2.0* API Support was added to Chrome, Firefox, and Opera.
  - Provides a new rendering context, and supports objects for the HTML5 canvas element
  - *HTML5* was enabled for all users by default in Chrome
  - Adobe Flash Player was disabled and only allowed to run with specific user permissions.
  - More than 100 fingerprinting APIs were found
- September 2017
  - *NetworkInformation* API was added to Chrome
  - Provides information about the connection a device is using to communicate with the network
- April 2017
  - *BaseAudioContext* API was added to Firefox
  - Acts as a base definition for online and offline audio-processing graphs

Analyzing Spikes in the Graph

- These APIs exist in our suspicious fingerprinting APIs list.
- Surprisingly, some of these APIs were not even mentioned directly in Chrome and Firefox release notes.
- Even a minor feature addition to a browser, which might not have been discussed in the release notes, could include fingerprintable APIs.

Incognito and Private Window Mode

- There is a small difference between total number of features in normal mode and incognito mode
  - Chrome 80's normal mode has 11,946 features.
  - Chrome 80's incognito mode has 11,936 features.
- Every fingerprinting API exists on both normal and incognito mode

- Incognito and Private Window mode have no impact on fingerprintability

Unique Features Set

- For each browser version in our study, we created a Feature Set
  - Includes all the extracted features
- There exist no two browsers that possess the same feature set
  - Only by looking at the feature set, each browser version is uniquely fingerprintable.
  - Add, remove, and re-adding features make them more fingerprintable
- Feature sets have become more similar recently.

Prior Work

- Multiple works on browser fingerprinting
  - Synder et al. Uses same method of feature collection. They measure the feature usage among the Alexa top websites.
  - Chenxiong et al. Propose a debloating framework for the browser that removes unused features
  - Eckersley. How a unique fingerprint is formed by combining wide range of browser properties
  - Cao et al, Olejnik et al, Nikiforakis et al, Mowery et al. Study on a single fingerprinting method

- Our aim was to collect data and analyze the trends to see whether we are becoming better at managing browser fingerprinting
- No study have looked at popular browsers historically and have attempted to determine how their fingerprintability has evolved over the years

Summary

- We extracted every browser feature in all browser versions using the browser APIs. We created a list of fingerprinting APIs based on different prior works.
- We show that all major Mozilla Firefox, Google Chrome, and Opera browser versions between 2016 until 2020 are uniquely fingerprintable based exclusively on the presence or absence of browser features
- We conclude that incognito mode has no impact on fingerprintability
- We analyze Mozilla Firefox, Google Chrome, and Opera and report major differences between feature introduction and removal trends.
- Although Chrome and Opera are both based upon Chromium and share the same codebase, there are still differences in their feature introduction and removal patterns.
    - But this shared codebase makes them very similar in our fingerprintability analysis.

Questions?