FP-Scanner:
The privacy implications of browser fingerprint inconsistencies

A. Vastel, P. Laperdrix, W. Rudametkin, R. Rouvoy
Browser fingerprinting in a nutshell

Stateless tracking technique

Combination of attributes from the browser:

**User agent:** “Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.87 Safari/537.36”

**Screen resolution:** “1280x720x24”

**Canvas:**

```
Cwm fjordbank glyphs vext quiz, 😊
```
Defense against fingerprinting

Different strategies:

- **Script blocking**: break collection
- **Attribute blocking**: decrease entropy
- **Attribute switching with pre-existing values**: break stability
- **Attribute blurring**: break stability

Different kinds of tools: browser extensions, forked browsers
Detecting countermeasures (1)

Fingerprinters may try to **detect countermeasures**:

- Augur
- FingerprintJS2
- Security fingerprinting scripts

Can be used as another fingerprinting attribute

```
"is":{
  "blockingAds":true,
  "blockingCookies":false,
  "blockingJava":true,
  "spoofed":true,
  "usingDoNotTrack":false,
  "incognito":false,
  "tor":false,
  "bot":false
}
```
Detecting countermeasures (2)

Use inconsistencies introduced by the countermeasure (Nikiforakis2013)

Example with a naive user agent spoofer:

- Real configuration: **Linux** with **Firefox**
  - `navigator.userAgent` = `Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36`
  - `navigator.platform` = **Linux x86_64**

The user agent says **Windows**, the platform says **Linux**
FP-Scanner

Verify if attributes of a fingerprint have been modified

Extend to all kinds of countermeasures

Use **inconsistencies** introduced by countermeasures

Split into 4 components:

- OS, browser, device, canvas
OS inconsistencies

Verify OS extracted from the user agent with:

- `Navigator.platform`
- `WebGL`

<table>
<thead>
<tr>
<th>OS</th>
<th>Vendor</th>
</tr>
</thead>
<tbody>
<tr>
<td>MacOS</td>
<td>Intel, ATI</td>
</tr>
<tr>
<td>Android</td>
<td>Qualcomm, ARM, Imagination</td>
</tr>
</tbody>
</table>
Browser inconsistencies (1)

Errors may be browser dependent:

Firefox

```
{
    depth: 108421,
    errorMessage: "too much recursion",
    errorName: "InternalError",
    errorStackSize: 6912
}
```

Chrome

```
{
    depth: 11416,
    errorMessage: "Maximum call stack size exceeded",
    errorName: "RangeError",
    errorStackSize: 1723
}
```
Browser inconsistencies (2)

Browser features: depends on **browser** and **version**

Function representation: `eval.toString()`

- **Safari** and **Firefox** ➔ "function eval() {
  
  [native code]

  }
"

- **Chrome** ➔ "function eval() { [native code] }"
Device inconsistencies

Is it really a **computer** or a **smartphone**?

Test the presence of events/sensors:

- Mouse on a phone: **onmousemove**
- Smartphone with no **accelerometer**
Canvas inconsistencies (1)

High entropy: depends on the device, browser, OS

High stability: important for tracking
Canvas inconsistencies (2)

A human can detect a visual difference between the 2 canvas

Constraints when defining the canvas:

- Background should be transparent
- There should not be isolated pixels
- Pixels in the rectangle should be (255, 102, 0, 100)

Verify if `toDataURL` and `getImageData` overridden:

```javascript
HTMLCanvasElement.prototype.toDataURL.toString();
```
Evaluation

Evaluation using 7 countermeasures:

- Canvas defender, Canvas FP Block, FP-Random (Canvas)
- Random Agent Spoofer, User agent spoofers
- Firefox protection, Brave

Compare with FingerprintJS2/Augur: verify OS, screen resolution, device, browser

Collect fingerprints with and without countermeasures from multiple devices
# Results

<table>
<thead>
<tr>
<th>Countermeasure</th>
<th>Accuracy FP-Scanner</th>
<th>Accuracy FP-JS2 / Augur</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Agent Spoofers</td>
<td>1.0</td>
<td>0.55</td>
</tr>
<tr>
<td>User agent spooers</td>
<td>1.0</td>
<td>0.86</td>
</tr>
<tr>
<td>Canvas Defender</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Firefox protection</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Canvas FP Block</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>FP-Random</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Brave</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>No countermeasure</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Tests failed by countermeasures

Random Agent Spoofeer: No accelerometer, `navigator.vendor` overridden

Canvas extensions and FP-Random: Pixels and `toDataURL` overridden

Brave: `navigator.mediaDevices.enumerateDevices`

Firefox fingerprinting protection: WebGL and media queries
Recovering ground values

Infer the real nature of the device: OS, browser + version

Recovering the OS: combine plugin extensions, WebGL, media queries, fonts

Recovering the browser:

- **Family:** `eval.toString().length` and `navigator.productSub`
- **Version:** Modernizr features

Infer real OS and browser family, but not the precise version
Privacy implications

**Discrimination:** similar to what happens with anti-adblockers

**Trackability:** can make the **user more easily trackable** (multiple factors):

1. Identify the countermeasure
2. Number of users
3. Ability to recover original values
4. Information leaked

**Does the anonymity gain provided by the countermeasure outweigh the anonymity loss caused by its detection?**
Example: Canvas Defender (1)

Chrome and Firefox extension: \( \approx 25k \) users

Randomize canvas by adding noise

Override toDataURL and getImageData

Genuine Canvas:

Modified Canvas:
Example: Canvas Defender (2)

```javascript
'function () {
    var width = this.width;
    var height = this.height;
    var context = this.getContext("2d");
    var imageData = context.getImageData(0, 0, width, height);
    for (var i = 0; i < height; i++) {
        ...
    }
    context.putImageData(imageData, 0, 0);
    showNotification();
    return old.apply(this, arguments);
}
```
Example: Canvas Defender (3)

Clone original `toDataURL` before Canvas Defender executes its code

```javascript
const getOriginalFunction = Function.prototype.call.bind(
  Function.prototype.bind,
  Function.prototype.call
);
const originalToDataURL = getOriginalFunction(HTMLCanvasElement.prototype.toDataURL);
```

Execute original function after `DOMContentLoaded` so that emojis are rendered correctly
Example: Canvas Defender (4)

Generate **random noise vector** \((r, g, b, a)\)

- Add noise component to each pixel

Detect when Canvas Defender code is added to the DOM (MutationObserver):

- **Extract the parameters of the function**, i.e. the noise vector
Example: Canvas Defender (5)

Canvas Defender can be identified

Small number of users \(~25k\)

→ Being detected with Canvas Defender is discriminative in itself

Can recover original canvas value

Leaks a potentially stable identifier (noise vector)
Conclusion

Fingerprinters can detect countermeasures using inconsistencies

Privacy implications:

- Discrimination
- Tracking

Same techniques could also be used to detect extensions with different settings