Detection of Browser Fingerprinting
by Static JavaScript Code Classification

Sjors Haanen & Tim van Zalingen

UvA

February 6, 2018

Supervisors (KPMG):
Aidan Barrington & Ruben de Vries

Research Project 82
Figure 1: Third party cookies

source: Mozilla - Lightbeam for Firefox
Browser fingerprinting

- Browser settings
- Hardware characteristics
- OS characteristics

\{\text{Unique fingerprint}\}

- Stateless
- Often even unnoticed by user
- Recent study could uniquely identify 89.4% out of 118,934 browsers\(^1\)

\(^1\)Laperdrix, Pierre 2017.
### Table 1: Excerpt fingerprinting results from https://amiunique.org

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Similarity ratio</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>User agent</td>
<td>&lt;0.1%</td>
<td>&quot;Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:58.0) Gecko...&quot;</td>
</tr>
<tr>
<td>Accept</td>
<td>54.78%</td>
<td>&quot;text/html,application/xhtml+xml,application/xhtml+xml,application/...&quot;</td>
</tr>
<tr>
<td>Content encoding</td>
<td>40.54%</td>
<td>&quot;gzip, deflate, br&quot;</td>
</tr>
<tr>
<td>Content language</td>
<td>27.53%</td>
<td>&quot;en-US,en;q=0.5&quot;</td>
</tr>
<tr>
<td>List of plugins</td>
<td>25.61%</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>Platform</td>
<td>10.64%</td>
<td>&quot;Linux x86_64&quot;</td>
</tr>
<tr>
<td>Cookies enabled</td>
<td>79.63%</td>
<td>&quot;yes&quot;</td>
</tr>
<tr>
<td>Do Not Track</td>
<td>30.51%</td>
<td>&quot;yes&quot;</td>
</tr>
<tr>
<td>Timezone</td>
<td>20.66%</td>
<td>&quot;-60&quot;</td>
</tr>
<tr>
<td>Screen resolution</td>
<td>21.29%</td>
<td>&quot;1920x1080x24&quot;</td>
</tr>
</tbody>
</table>
Defences against browser fingerprinting

- Disable functionality
- N:1 - Many Browsers, One Configuration (Tor)
- 1:N - One Browser, Many Configurations
  - Randomise data per request/session
Motivation

- Privacy
- Existing detection and prevention solutions often criticised
Prior work

Previous attempts to detect fingerprinting:
- Blacklists\textsuperscript{2}
- Dynamic analysis: detection at runtime\textsuperscript{3}
- Static analysis: counting\textsuperscript{4}

\textsuperscript{2}Kontaxis, Georgios and Chew, Monica 2015.
\textsuperscript{3}Acar, Gunes and Juarez, Marc and Nikiforakis, Nick and Diaz, Claudia and Gürses, Seda and Piessens, Frank and Preneel, Bart 2013; FaizKhademi, Amin and Zulkernine, Mohammad and Weldemariam, Komminist 2015.
\textsuperscript{4}Rausch, Michael and Good, Nathan and Hoofnagle, Chris Jay 2014.
Can the action of browser fingerprinting be detected before execution by analysing JavaScript code with machine learning?
Figure 2: Process of analysing JavaScript (JS) source code for a given set of websites to find fingerprinting practices
Method overview

Gathering
Collect sets of scripts

Processing
Deobfuscation
Member expressions expansion

Detection
Count suspicious calls
SVM classification
Collect sets of scripts

Predefined sets (by manual search):
- Set of 12 fingerprinting scripts
- Set of 20 non-fingerprinting scripts
Method overview

Gathering
- Collect sets of scripts

Processing
- Deobfuscation
- Member expressions expansion

Detection
- Count suspicious calls
- SVM classification

Sjors Haanen & Tim van Zalingen (UvA)

RP82: Browser Fingerprinting
February 6, 2018
Deobfuscation: The problem

eval(function(p,a,c,k,e,d){e=function(c){return c.toString(36)};
if(!'' .replace(/\^/,String)) {while(c--){d[c.toString(a)]=k[c] ||c.toString(a)} k=[function(e){return d[e]}];
e=function(){return '\w+'};
c=1;while(c--){if(k[c]){p=p.replace(new RegExp('
\b'+e(c)+'\b','g'),k[c])}} return p}('0 1=3;8 4(){0 a=1.2;
0 b=a;0 5=b.6;0 7=1.9}')',12,12,'var nav|plugins|navigator|fingerprint|c|length|d|function|userAgent|'.split('|'),0,{})

Figure 3: An example of JS code obfuscated by www.danstools.com/javascript-obfuscate/

Who can tell us what this piece of code does?
Deobfuscation: JSBeautifier

Requirements:
- Counter obfuscation
- Counter minification
- Counter packing

```javascript
var nav = navigator;
function fingerprint () {
    var a = nav.plugins;
    var b = a;
    var c = b.length;
    var d = nav.userAgent
}
```

Figure 4: The JS code in figure 3 deobfuscated by http://jsbeautifier.org/
Method overview

1. **Gathering**
   - Collect sets of scripts

2. **Processing**
   - Deobfuscation
   - Member expressions expansion

3. **Detection**
   - Count suspicious calls
   - SVM classification
Expanding member expressions: The problem

```javascript
var nav = navigator;
function fingerprint() {
    var a = nav.plugins;
    var b = a;
    var c = b.length;
    var d = nav.userAgent;
}
```

Figure 5: Example JS code with split member expressions

Figure 6: Expanded member expressions for the code in figure 5
Expanding member expressions: Abstract Syntax Tree (AST)

- Parse code
- Traverse AST
- Analyse scope

```javascript
var nav = navigator;
function fingerprint() {
  var a = nav.plugins;
}
```

**Figure 7:** Example JS code with split member expressions

**Figure 8:** The Abstract Syntax Tree of the code in figure 7
Method overview

Gathering
- Collect sets of scripts

Processing
- Deobfuscation
  - Member expressions expansion

Detection
- Count suspicious calls
  - SVM classification
Count suspicious calls

Counting calls in processed files aggregated per domain

Examples of suspicious JS calls:

- navigator.userAgent
- navigator.plugins.name
- navigator.javaEnabled()
- window.screen.colorDepth
- Date().getTimezoneOffset()
Figure 9: Comparing different JS calls that can be used as a feature to differentiate scripts
Method overview

Gathering
- Collect sets of scripts

Processing
- Deobfuscation
  - Member expressions expansion
- Count suspicious calls
  - SVM classification
Support Vector Machine (SVM)

- Supervised learning methods
- Classification
- Relevant advantages:
  - Effective in high dimensional spaces
  - Effective with more dimensions than samples
- Avoid over-fitting with small number of samples
Occurences in JS code per website. Linear classification by SVM.

Figure 10: SVM Classification example for two features
Figure 11: SVM Classification example for two features. These two features are not easily distinguishable.
Support Vector Machine: Prevent overfitting

- Partition data into training and test set
- Cross-validation
- Stratified k-fold preserves positive and negative ratio

Figure 12: Visualised example of k-fold cross-validation with k=4
(source: Wikipedia - Cross-validation (statistics))
Results: Full dimensional classification

Figure 13: Receiver Operating Characteristic curve to illustrate the performance of the classifier
$F_1$-score=0.80
Observable difference, SVM can detect fingerprinting scripts

- Combining features and using a classifier improves on earlier research
- Future implementation of proposed method might aid in detection
- False positives
Future work

- Refine list of suspicious JS calls
- Include other signs of fingerprinting in the analysis, e.g.:
  - Hashing values
  - Sending fingerprintable data to a remote server
- Bigger dataset
- Other machine learning algorithms

