Fast and efficient Browser Identification with JavaScript Engine Fingerprinting

Martin Mulazzani, Philipp Reschl, Manuel Leithner, Markus Huber, Edgar Weippl

SBA Research
Vienna, Austria

secure
sba-research.org
Outline

Motivation & Background

JavaScript Engine Fingerprinting
    Methodology
    Minimal Fingerprints
    Decision Trees

Evaluation
    Evaluation - Tor Browser Bundle
    Evaluation - Survey
Motivation

Browser Identification:

- Accurately identify the browser used by the client
- Webserver point-of-view
- Motivated by *nmap* for TCP/IP fingerprinting
- Limitations of UserAgent string:
  - Can be set arbitrarily
  - Not a security feature

Different use cases:

- Detect UserAgent string manipulations
- Detect session hijacking
- Browser-specific malware
Motivation

Browser Identification:
- Accurately identify the browser used by the client
- Webserver point-of-view
- Motivated by *nmap* for TCP/IP fingerprinting
- Limitations of UserAgent string:
  - Can be set arbitrarily
  - Not a security feature

Different use cases:
- Detect UserAgent string manipulations
- Detect session hijacking
- Browser-specific malware
Browser Market

Browser market currently very competitive:

- Man-years of development time
- Fight for market shares, especially smartphones
- Become more & more powerful (e.g., Cloud computing, HTML5, ...)
- New features:
  - JIT, GPU rendering, remote rendering, Sandboxing
  - Mostly **performance** or **security**
Browser Market :)
Methodology

Our approach:
- Use JavaScript (ECMAScript 5.1) conformance tests
  - test262 - http://test262.ecmascript.org
  - Sputnik - http://sputnik.googlelabs.com
- More than 11,000 test cases
- Javascript engines fail at different test cases

In the future:
- Enhance session security
  - by locking session to specific browser version
- Increase user privacy
  - by detecting (attacking) fingerprinting
Methodology

Our approach:

- Use JavaScript (ECMAScript 5.1) conformance tests
  - test262 - http://test262.ecmascript.org
  - Sputnik - http://sputnik.googlelabs.com
- More than 11,000 test cases
- Javascript engines fail at different test cases

In the future:

- **Enhance session security**
  - by locking session to specific browser version
- **Increase user privacy**
  - by detecting (attacking) fingerprinting
Methodology

Our approach:

▶ Use JavaScript (ECMAScript 5.1) conformance tests
  ▶ test262 - http://test262.ecmascript.org
  ▶ Sputnik - http://sputnik.googlelabs.com
▶ More than 11,000 test cases
▶ Javascript engines fail at different test cases

In the future:

▶ **Enhance session security**
  ▶ by locking session to specific browser version
▶ **Increase user privacy**
  ▶ by detecting (attacking) fingerprinting
Related Work

Recent paper by Mowery et.al, W2SP 2011
- Use 39 Javascript benchmarks e.g., Sunspider or V8 Benchmark Suite
- Generate normalized fingerprint based on time pattern
- On average 190 seconds runtime

Our approach:
- Takes less then 200ms (3 orders of magnitude faster)
- not stalling the CPU noticeably
- Few hundred lines of Javascript max.
- Collected $> 150$ OS and browser combinations
Related Work

Recent paper by Mowery et.al, W2SP 2011
▶ Use 39 Javascript benchmarks e.g., Sunspider or V8 Benchmark Suite
▶ Generate normalized fingerprint based on time pattern
▶ On average 190 seconds runtime

Our approach:
▶ Takes less than 200ms (3 orders of magnitude faster)
▶ not stalling the CPU noticeably
▶ Few hundred lines of Javascript max.
▶ Collected $> 150$ OS and browser combinations
Related Work

Other related work:

▶ EFF’s Panopticlick, PETS 2010
▶ Mowery et.al, W2SP 2012
  ▶ uses novel HTML5 features and WebGL rendering
▶ Upcoming paper on HTML5 and CSS3 features (ARES 2013)
Please click on the Start button to start the test. Once you start the test you may pause the test anytime by clicking on the Pause button. You can click on the Results tab once the test is completed or after pausing the test. The Reset button is for restarting the test run.

Tests To Run: 11181 | Total Tests Ran: 4764 | Pass: 4748 | Fail: 16 | Failed To Load: 0

Running Test: 15.2.3.3-4-43

- **S7.8.4_A7.2_T6**: HexDigit :: A
  - **Fail**
- **S7.8.4_A7.2_T3**: HaxDigit :: 1
  - **Fail**
- **S7.8.4_A7.2_T4**: HexDigit :: A
  - **Fail**
- **S7.8.4_A7.2_T5**: HaxDigit :: 1
  - **Fail**
- **S7.8.4_A7.2_T6**: HexDigit :: A
  - **Fail**
- **10.4.2.1-1gs**: Strict Mode - eval code cannot instantiate variable in the variable environment of the calling context that invoked the eval if the code of the calling context is strict code
  - **Fail**
- **S10.4.2.1_A1**: Strict indirect eval should not leak top level declarations into the global scope
  - **Fail**
- **S15.1.2.2_A5.1_T1**: Check if parseInt still accepts octal
  - **Fail**
- **S15.10.2.12_A1_T1**: WhiteSpace
  - **Fail**
- **S15.10.2.12_A2_T1**: WhiteSpace
  - **Fail**
- **S15.12.2_A1**: Tests that JSON.parse treats "__proto__" as a regular property name
  - **Fail**

Test Suite Ver.: ES5.1 | Test Suite Date: 2012-01-16
<table>
<thead>
<tr>
<th>Browser</th>
<th>Win 7</th>
<th>WinXP</th>
<th>Mac OS X</th>
<th>Browser</th>
<th>Win 7</th>
<th>WinXP</th>
<th>Mac OS X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefox 3.6.26</td>
<td>3955</td>
<td>3955</td>
<td>3955</td>
<td>Chrome 8</td>
<td>1022</td>
<td>1022</td>
<td>1022</td>
</tr>
<tr>
<td>Firefox 4</td>
<td>290</td>
<td>290</td>
<td>290</td>
<td>Chrome 10</td>
<td>715</td>
<td>715</td>
<td>715</td>
</tr>
<tr>
<td>Firefox 5</td>
<td>264</td>
<td>264</td>
<td>264</td>
<td>Chrome 11</td>
<td>489</td>
<td>489</td>
<td>489</td>
</tr>
<tr>
<td>Firefox 6</td>
<td>214</td>
<td>214</td>
<td>214</td>
<td>Chrome 12</td>
<td>449</td>
<td>449</td>
<td>—</td>
</tr>
<tr>
<td>Firefox 7</td>
<td>190</td>
<td>190</td>
<td>190</td>
<td>Chrome 13</td>
<td>427</td>
<td>427</td>
<td>—</td>
</tr>
<tr>
<td>Firefox 12</td>
<td>165</td>
<td>165</td>
<td>165</td>
<td>Chrome 14</td>
<td>430</td>
<td>430</td>
<td>430</td>
</tr>
<tr>
<td>Firefox 15</td>
<td>161</td>
<td>161</td>
<td>161</td>
<td>Chrome 16</td>
<td>420</td>
<td>420</td>
<td>420</td>
</tr>
<tr>
<td>Firefox 17</td>
<td>171</td>
<td>171</td>
<td>171</td>
<td>Chrome 17</td>
<td>210</td>
<td>210</td>
<td>210</td>
</tr>
<tr>
<td>Firefox 19</td>
<td>191</td>
<td>191</td>
<td>191</td>
<td>Chrome 18</td>
<td>35</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>IE 6 (Sputnik)</td>
<td>—</td>
<td>468</td>
<td>—</td>
<td>Chrome 19</td>
<td>18</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>IE 8 (Sputnik)</td>
<td>—</td>
<td>473</td>
<td>—</td>
<td>Chrome 21</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>IE 9</td>
<td>611</td>
<td>—</td>
<td>—</td>
<td>Chrome 23</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>IE 10</td>
<td>7</td>
<td>—</td>
<td>—</td>
<td>Chrome 25</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Opera 11.52</td>
<td>3827</td>
<td>3827</td>
<td>3827</td>
<td>Safari 5.0.5</td>
<td>777</td>
<td>1585</td>
<td>1513</td>
</tr>
<tr>
<td>Opera 11.64</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>Safari 5.1</td>
<td>777</td>
<td>853</td>
<td>—</td>
</tr>
<tr>
<td>Opera 12.02</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>Safari 5.1.2</td>
<td>777</td>
<td>777</td>
<td>776</td>
</tr>
<tr>
<td>Opera 12.14</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>Safari 5.1.7</td>
<td>548</td>
<td>548</td>
<td>547</td>
</tr>
<tr>
<td>Browser</td>
<td>OS</td>
<td>Device</td>
<td># of fails</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>------------</td>
<td>------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Safari</td>
<td>iOS 5.1.1</td>
<td>iPhone 4S</td>
<td>988</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Safari</td>
<td>iOS 6.1.2</td>
<td>iPhone 4</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browser</td>
<td>Android 2.2</td>
<td>GalaxyTab</td>
<td>2130</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browser</td>
<td>Android 2.3.7</td>
<td>HTC Desire</td>
<td>1328</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browser</td>
<td>Android 4.0.3</td>
<td>GalaxyTab2</td>
<td>588</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browser</td>
<td>Android 4.0.4</td>
<td>Nexus S</td>
<td>591</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browser</td>
<td>Android 4.1.2</td>
<td>Nexus S</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chrome 18</td>
<td>Android 4.0.3</td>
<td>GalaxyTab2</td>
<td>46</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firefox 19</td>
<td>Android 4.0.3</td>
<td>GalaxyTab2</td>
<td>191</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Distinguish Browsers

Random subset of test262 test cases:

<table>
<thead>
<tr>
<th>Web Browser</th>
<th>15.4.4.4-5-c-i-1</th>
<th>13.0-13-s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opera 11.61</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Firefox 10.0.1</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Internet Explorer 9</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Chrome 17</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Web Browser</th>
<th>S15.2.3.6_A1</th>
<th>10.6-7-1</th>
<th>S10.4.2.1_A1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opera 11.61</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Firefox 10.0.1</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Internet Explorer 9</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Chrome 17</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
Two Methods

Propose two different methods:

1. Minimal fingerprints
   ▶ Find out if a browser is lying about it’s UserAgent

2. Iterative decision trees
   ▶ Find browser with no a-priory knowledge

Sharing is caring:

▶ Will release code & collected dataset
▶ Lost due to hardware failure
▶ Drop me an email for current version
▶ Always test your backups!
Two Methods

Propose two different methods:

1. Minimal fingerprints
   ▶ Find out if a browser is lying about it’s UserAgent

2. Iterative decision trees
   ▶ Find browser with no a-priory knowledge

Sharing is caring:

▶ Will release code & collected dataset
▶ Lost due to hardware failure
▶ Drop me an email for current version
▶ Always test your backups!
Minimal Fingerprints

Goal: Determine minimal fingerprints

1. Define the testset (=set of browsers)
2. Collect failed test cases
3. Calculate minimal fingerprints
4. For every client: Run fingerprints

Result: If browser version $\in$ testset: confirm browser version

“Mind the gap:”

- Propably not for every testset solvable
- Can become “big”
Minimal Fingerprints

Goal: Determine minimal fingerprints

1. Define the testset (=set of browsers)
2. Collect failed test cases
3. Calculate minimal fingerprints
4. For every client: Run fingerprints

Result: If browser version $\in$ testset: confirm browser version

“Mind the gap:”

- Propably not for every testset solvable
- Can become “big”
Decision Trees

Goal: Minimize number of tests run at the client
1. Define the testset (=set of browsers)
2. Collect failed test cases
3. Calculate uniqueness of every failed test case
4. Build binary decision tree, iteratively
Result: Minimal path through decision tree for unknown browsers

Benefits:
- $O(\log n)$ instead of $O(n)$
- Thus even faster
- Can be used as first stage for minimal fingerprinting
Decision Trees

Goal: Minimize number of tests run at the client

1. Define the testset (=set of browsers)
2. Collect failed test cases
3. Calculate *uniqueness* of every failed test case
4. Build binary decision tree, iteratively

Result: Minimal path through decision tree for unknown browsers

Benefits:

- $O(\log n)$ instead of $O(n)$
- Thus *even faster*
- Can be used as first stage for minimal fingerprinting
Decision Trees

15.4.4.4-5-c-i-1

10.6-7-1

13.0-13-s

✅

✗

✗

✅

✗

✅

✗

✅
Evaluation - Tor Browser Bundle

Basics Tor:
- Internet anonymization network
- Hides a user’s real IP address
- Hundreds of thousands users every day
- Approx. 3000 servers run by volunteers

Tor Browser Bundle:
- Among other features: **Uniform UserAgent**
  - to increase size of the anonymity set
- Everything prepackaged (Tor, Vidalia, Firefox, ...)
- Runs without admin rights
Evaluation - Tor Browser Bundle

Basics Tor:
- Internet anonymization network
- Hides a user’s real IP adress
- Hundreds of thousands users every day
- Approx. 3000 servers run by volunteers

Tor Browser Bundle:
- Among other features: **Uniform UserAgent**
  - to increase size of the anonymity set
- Everything prepackaged (Tor, Vidalia, Firefox, ...)
- Runs without admin rights
Evaluation - Tor Browser Bundle

Uniform UserAgent:

- Tor - Mozilla/5.0 (Windows NT 6.1; rv:5.0)
  Gecko/20100101 Firefox/5.0
- Real - Mozilla/5.0 (X11; Linux x86_64; rv:9.0.1)
  Gecko/20111222 Firefox/9.0.1

Vulnerable to Javascript Engine Fingerprinting?

- Yes!
- Every Firefox > 3.5 can be easily distinguished
- Can harm user privacy and decrease anonymity set
- However, not a real attack on Tor
Evaluation - Tor Browser Bundle

Uniform UserAgent:

- Tor - Mozilla/5.0 (Windows NT 6.1; rv:5.0) Gecko/20100101 Firefox/5.0
- Real - Mozilla/5.0 (X11; Linux x86_64; rv:9.0.1) Gecko/20111222 Firefox/9.0.1

Vulnerable to Javascript Engine Fingerprinting?

- Yes!
- Every Firefox > 3.5 can be easily distinguished
- Can harm user privacy and decrease anonymity set
- However, not a real attack on Tor
# Evaluation - Tor Browser Bundle

<table>
<thead>
<tr>
<th>Version TBB</th>
<th>Browser</th>
<th>UserAgent</th>
<th>test262</th>
<th>exp. test262</th>
<th>Detectable</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.25-4</td>
<td>Firefox 17esr</td>
<td>Firefox 17</td>
<td>171</td>
<td>171</td>
<td>✗</td>
</tr>
<tr>
<td>2.3.25-2</td>
<td>Firefox 10esr</td>
<td>Firefox 10</td>
<td>172</td>
<td>172</td>
<td>✗</td>
</tr>
<tr>
<td>2.2.35-9</td>
<td>Firefox 12.0</td>
<td>Firefox 5.0</td>
<td>165</td>
<td>264</td>
<td>✓</td>
</tr>
<tr>
<td>2.2.35-8</td>
<td>Firefox 11.0</td>
<td>Firefox 5.0</td>
<td>164</td>
<td>264</td>
<td>✓</td>
</tr>
<tr>
<td>2.2.35-3</td>
<td>Firefox 9.0.1</td>
<td>Firefox 5.0</td>
<td>167</td>
<td>264</td>
<td>✓</td>
</tr>
<tr>
<td>2.2.33-2</td>
<td>Firefox 7.0.1</td>
<td>Firefox 5.0</td>
<td>190</td>
<td>264</td>
<td>✓</td>
</tr>
<tr>
<td>2.2.32-3</td>
<td>Firefox 6.0.2</td>
<td>Firefox 5.0</td>
<td>214</td>
<td>264</td>
<td>✓</td>
</tr>
<tr>
<td>2.2.30-2</td>
<td>Firefox 5.0.1</td>
<td>Firefox 5.0</td>
<td>264</td>
<td>264</td>
<td>✗</td>
</tr>
<tr>
<td>2.2.24-1</td>
<td>Firefox 4.0</td>
<td>Firefox 3.6.3</td>
<td>290</td>
<td>3956</td>
<td>✓</td>
</tr>
</tbody>
</table>
Evaluation - Survey

Tested our fingerprinting with a survey:

- 189 participants
- Open for a few weeks in Summer 2011
- 10 test cases per browser in testset
- Testset:
  - IE 8
  - IE 9
  - Chrome 10
  - Firefox 4

Ground truth:

- UserAgent String
- Manual identification by participant
Evaluation - Survey

Tested our fingerprinting with a survey:

- 189 participants
- Open for a few weeks in Summer 2011
- 10 test cases per browser in testset
- Testset:
  - IE 8
  - IE 9
  - Chrome 10
  - Firefox 4

Ground truth:

- UserAgent String
- Manual identification by participant
Evaluation - Survey

Performance:
- All files: 24 Kilobytes
- Fingerprints: (4x) 2.500-3.000 Bytes
- **90 ms** on average on PC
- **200 ms** on average on smartphone

Results:
- 175 out of 189 browsers covered by testset
  - 100 % detection rate
  - No false positives!
- 14 not covered were mostly smartphones
- 1 UserAgent manipulation discovered
Evaluation - Survey

Performance:
- All files: 24 Kilobytes
- Fingerprints: (4x) 2.500-3.000 Bytes
- 90 ms on average on PC
- 200 ms on average on smartphone

Results:
- 175 out of 189 browsers covered by testset
  - 100 % detection rate
  - No false positives!
- 14 not covered were mostly smartphones
- 1 UserAgent manipulation discovered
Thank you for your time!

Questions?

mmulazzani@sba-research.org