Fast and efficient Browser Identification with JavaScript Engine Fingerprinting

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

SBA Research
Vienna, Austria

Outline

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 Benachmark 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 Benachmark Suite
- Generate normalized fingerprint based on time pattern
- On average 190 seconds runtime

Our approach:

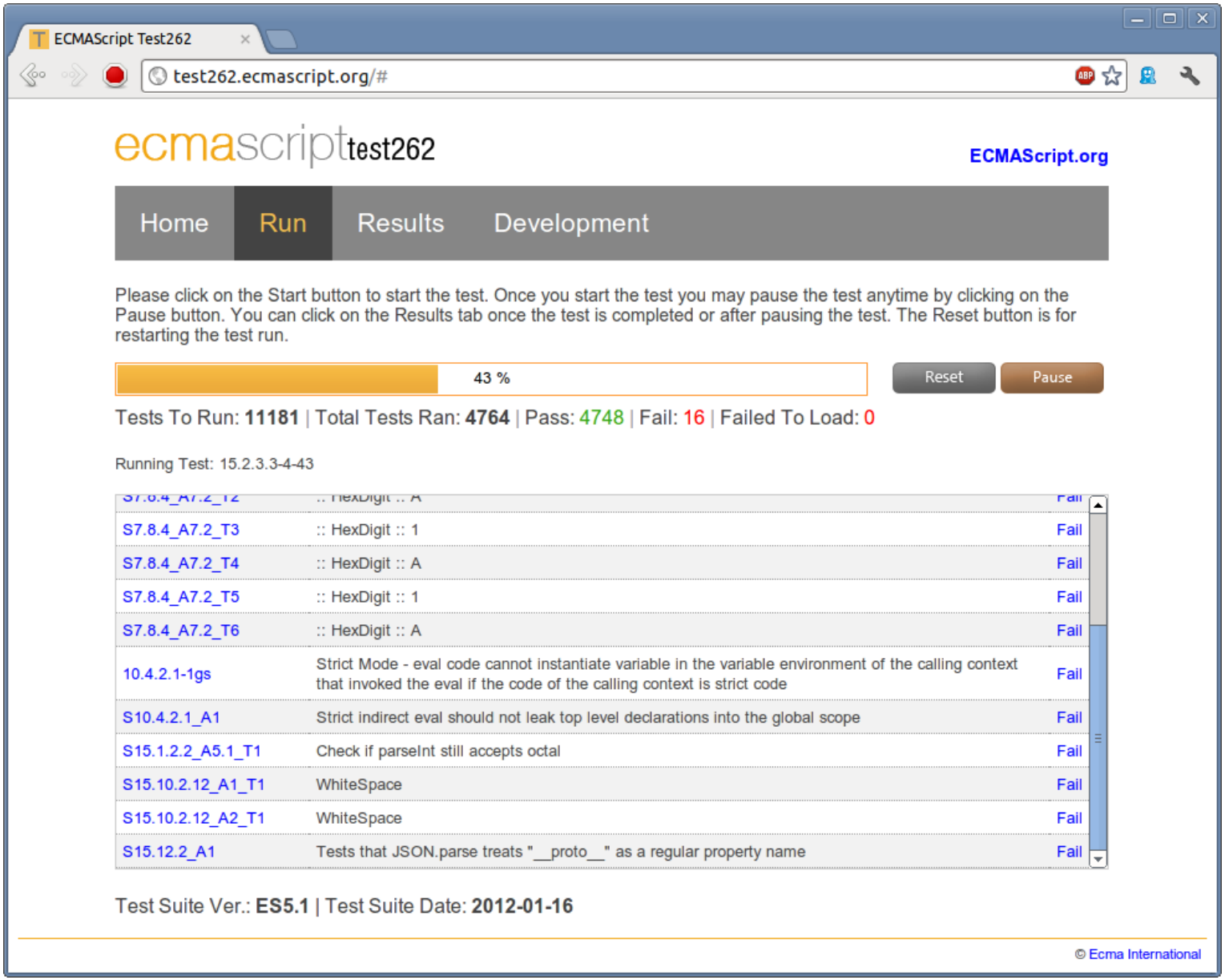
- Takes less then 200ms (3 orders of magnitude faster)
- not stalling the CPU noticeably
- Few hundred lines of Javascript max.
- Collected $>$ 150 OS and browser combinations

Related Work

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)

test262


ECMAScript Test262
test262.ecmascript.org/#
ecmascripttest262
ECMAScript.org
Home
Run
Results
Development
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.
43 %
Reset
Pause
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_T3 :: HexDigit :: 1 Fail
S7.8.4_A7.2_T4 :: HexDigit :: A Fail
S7.8.4_A7.2_T5 :: HexDigit :: 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
© Ecma International

test262: Browser - OS Combinations

Browser	Win 7	WinXP	Mac OS X
Firefox 3.6.26	3955	3955	3955
Firefox 4	290	290	290
Firefox 5	264	264	264
Firefox 6	214	214	214
Firefox 7	190	190	190
Firefox 12	165	165	165
Firefox 15	161	161	161
Firefox 17	171	171	171
Firefox 19	191	191	191
IE 6 (Sputnik)	—	468	—
IE 8 (Sputnik)	—	473	—
IE 9	611	—	—
IE 10	7	—	—
Opera 11.52	3827	3827	3827
Opera 11.64	4	4	4
Opera 12.02	4	4	4
Opera 12.14	9	9	9

Browser	Win 7	WinXP	Mac OS X
Chrome 8	1022	1022	1022
Chrome 10	715	715	715
Chrome 11	489	489	489
Chrome 12	449	449	—
Chrome 13	427	427	—
Chrome 14	430	430	430
Chrome 16	420	420	420
Chrome 17	210	210	210
Chrome 18	35	35	35
Chrome 19	18	18	18
Chrome 21	9	9	9
Chrome 23	10	10	10
Chrome 25	17	17	17
Safari 5.0.5	777	1585	1513
Safari 5.1	777	853	—
Safari 5.1.2	777	777	776
Safari 5.1.7	548	548	547

test262: Browser - OS Combinations

Browser	OS	Device	# of fails
Safari	iOS 5.1.1	iPhone 4S	988
Safari	iOS 6.1.2	iPhone 4	28
Browser	Android 2.2	GalaxyTab	2130
Browser	Android 2.3.7	HTC Desire	1328
Browser	Android 4.0.3	GalaxyTab2	588
Browser	Android 4.0.4	Nexus S	591
Browser	Android 4.1.2	Nexus S	23
Chrome 18	Android 4.0.3	GalaxyTab2	46
Firefox 19	Android 4.0.3	GalaxyTab2	191

Distinguish Browsers

Random subset of *test262* test cases:

Web Browser	15.4.4.4-5-c-i-1	13.0-13-s
Opera 11.61	✓	✗
Firefox 10.0.1	✓	✗
Internet Explorer 9	✗	✓
Chrome 17	✗	✗

Web Browser	S15.2.3.6_A1	10.6-7-1	S10.4.2.1_A1
Opera 11.61	✗	✗	✗
Firefox 10.0.1	✗	✓	✗
Internet Explorer 9	✗	✗	✓
Chrome 17	✓	✗	✓

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:

- $O(logn)$ 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

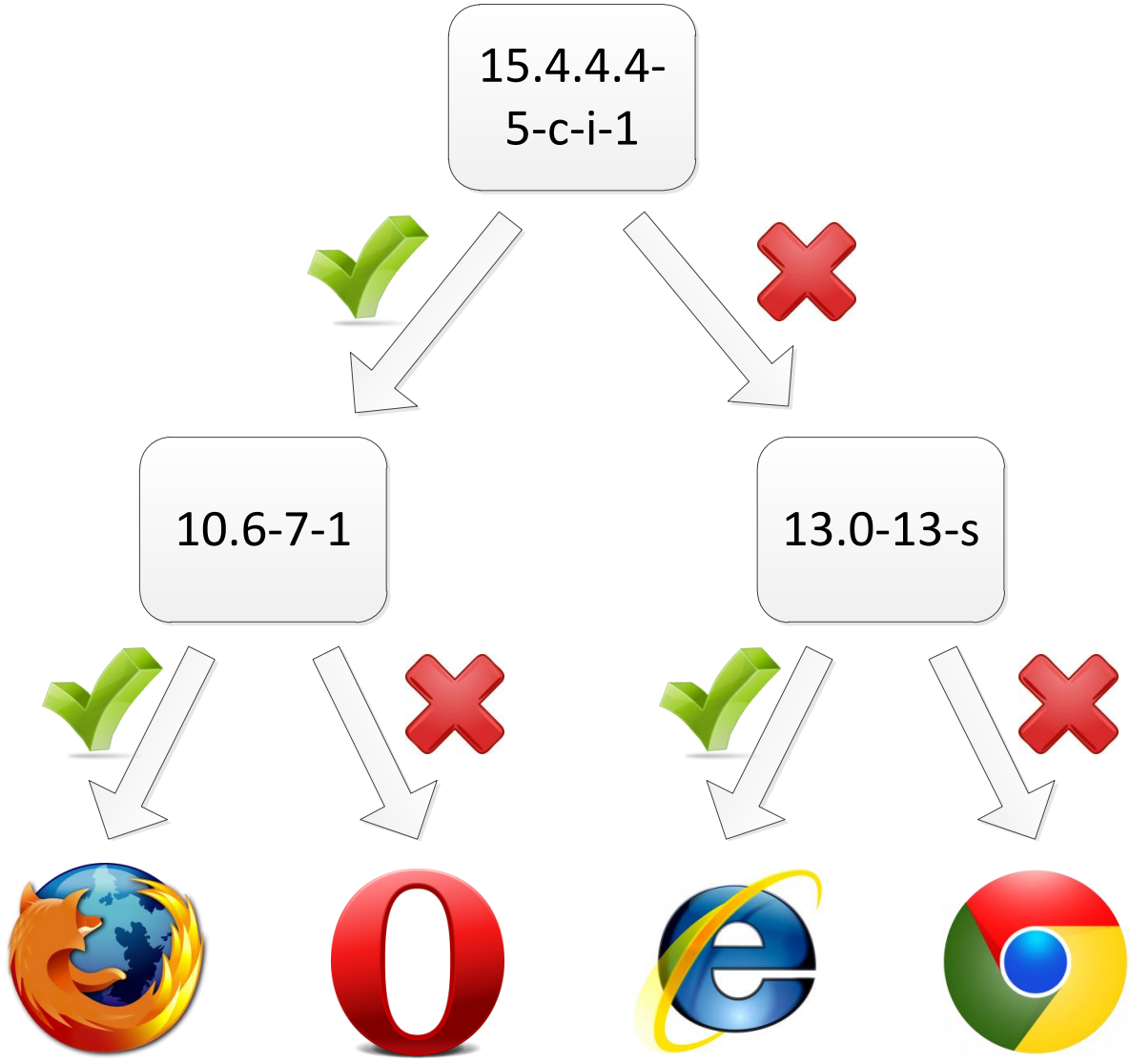
1. Define the testset (=set of browsers)
2. Collect failed test cases
3. Calculate *uniqueness* of every failed test case
4. Build binary decision tree, iteratively

Result: Minimal path through decision tree for unknown browsers

Benefits:

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

Decision Trees



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

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

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

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