浏览器模糊测试

Browser Fuzzing

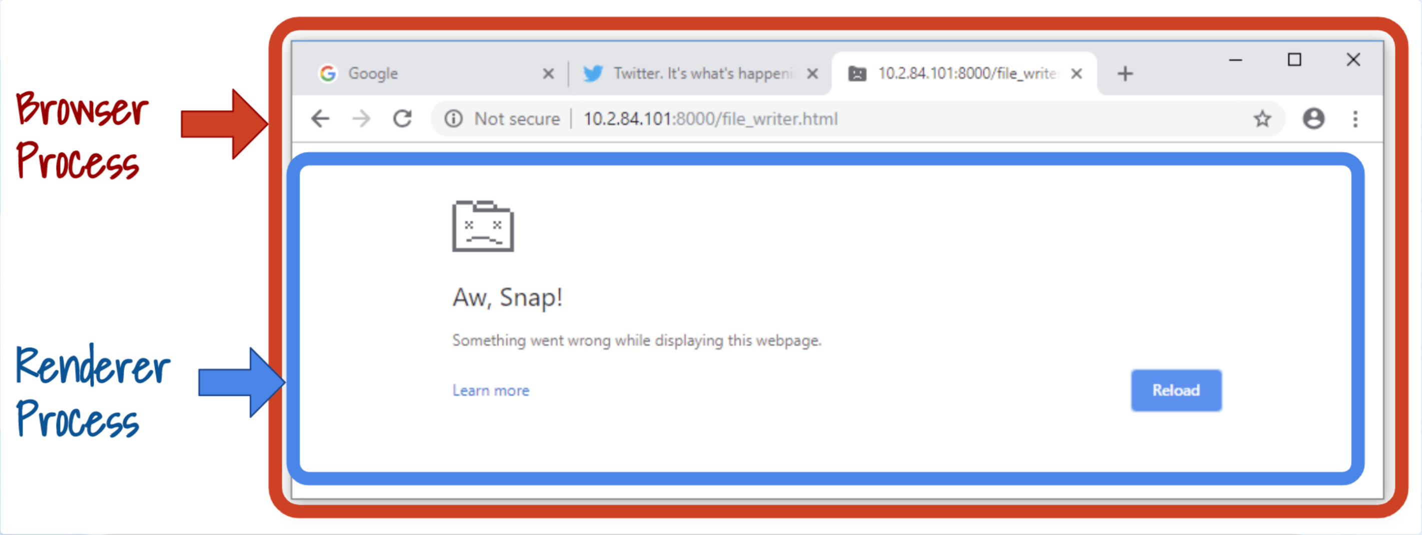
2024.12

郑力澜

Background： Browser Security

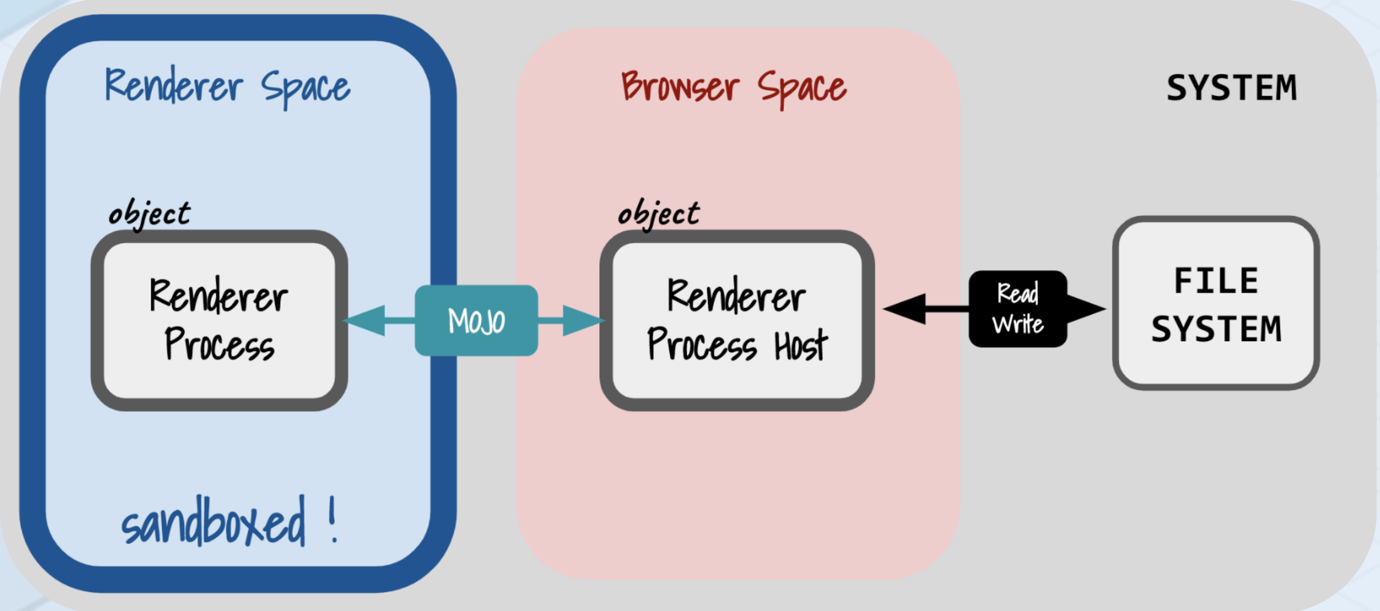
浏览器模型（e.g., Chrome）

- Renderer Process
  - 在sandbox内，权限较低
  - 通过ipc与Browser Process交互
- Browser Process
  - 权限较高，可以正常读写文件
  - system


Browser Process
Renderer Process
Google
Twitter. It's what's happeni
10.2.84.101:8000/file_write
Not secure
10.2.84.101:8000/file_writer.html
Learn more
Reload

Aw, Snap!

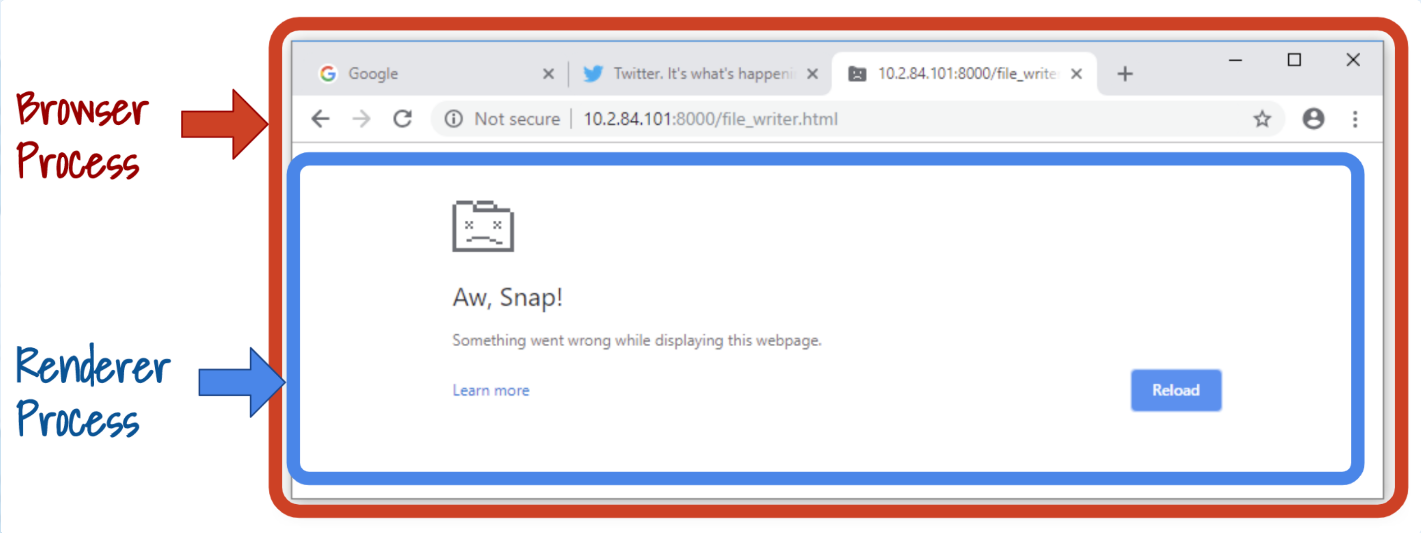
Something went wrong while displaying this webpage.



Background： Browser Security

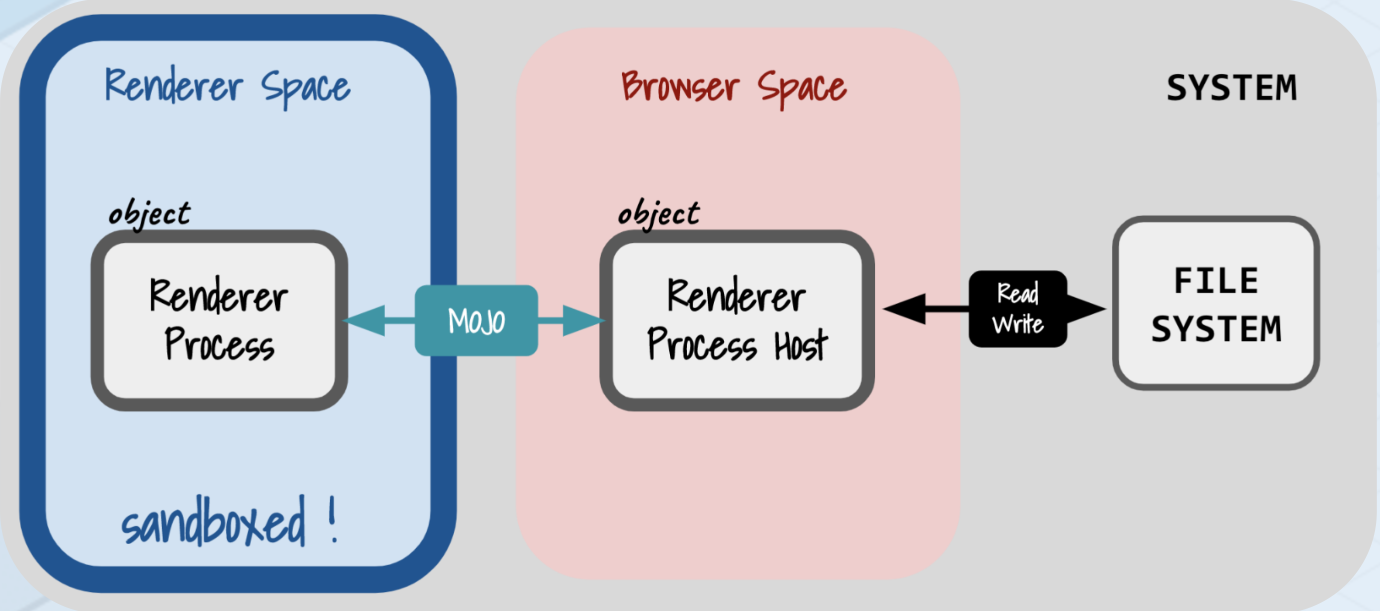
浏览器漏洞分类

- **RCE（Remote Code Execution）**
  - 在Renderer中执行任意代码
  - 利用内存漏洞
- Sandbox Escape
  - 在Renderer RCE基础上突破Sandbox限制
  - 利用IPC漏洞等..


Browser Process
Renderer Process
Google
Twitter. It's what's happeni
10.2.84.101:8000/file_write
Not secure
10.2.84.101:8000/file_writer.html
Learn more
Reload

Aw, Snap!

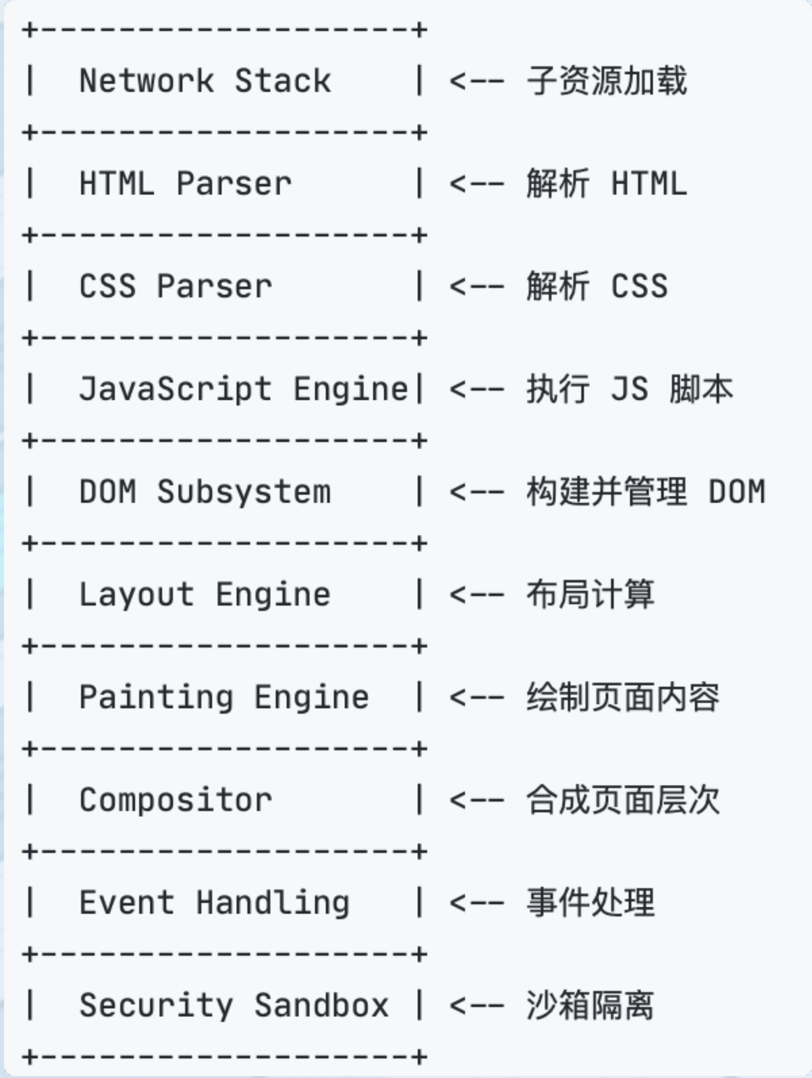
Something went wrong while displaying this webpage.



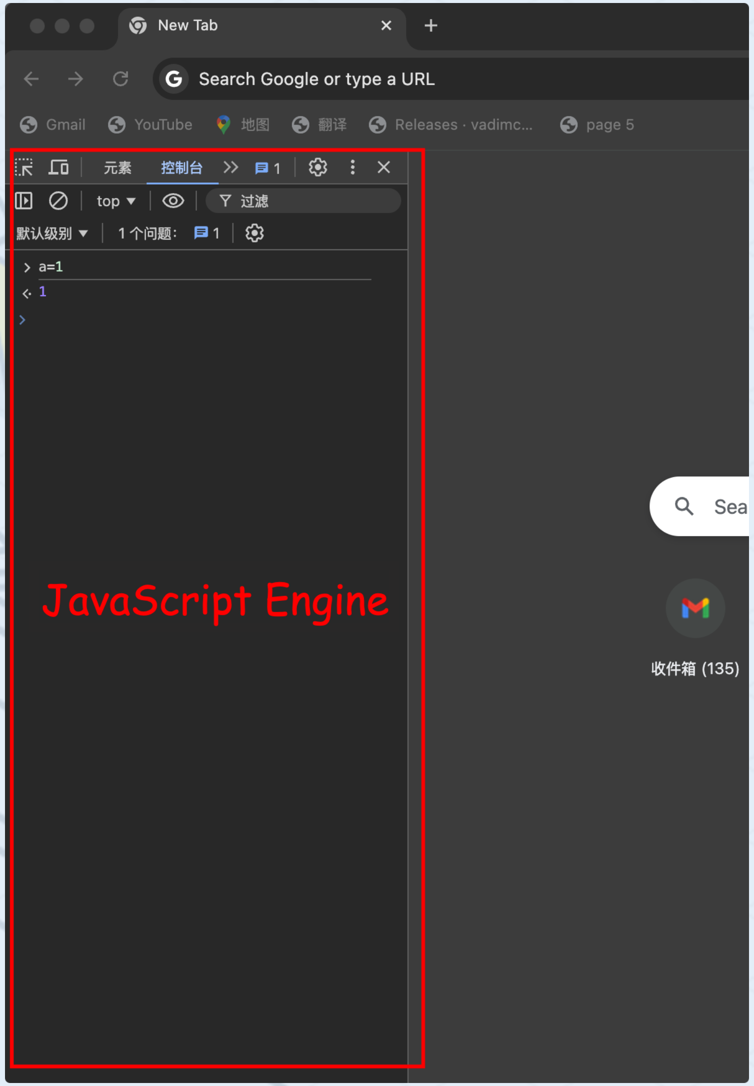
Background： Browser Security ——Renderer RCE

Renderer组件

- JavaScript Engine⭐
  - 类型混淆漏洞
  - OOB（Out-of-Bounds）
  - JIT
  - …
- HTML Parser
- CSS Parser
- …



```
+-------------------+
|  Network Stack    | <-- 子资源加载
+-------------------+
|  HTML Parser      | <-- 解析 HTML
+-------------------+
|  CSS Parser       | <-- 解析 CSS
+-------------------+
|  JavaScript Engine| <-- 执行 JS 脚本
+-------------------+
|  DOM Subsystem    | <-- 构建并管理 DOM
+-------------------+
|  Layout Engine    | <-- 布局计算
+-------------------+
|  Painting Engine  | <-- 绘制页面内容
+-------------------+
|  Compositor       | <-- 合成页面层次
+-------------------+
|  Event Handling   | <-- 事件处理
+-------------------+
|  Security Sandbox | <-- 沙箱隔离
+-------------------+
```



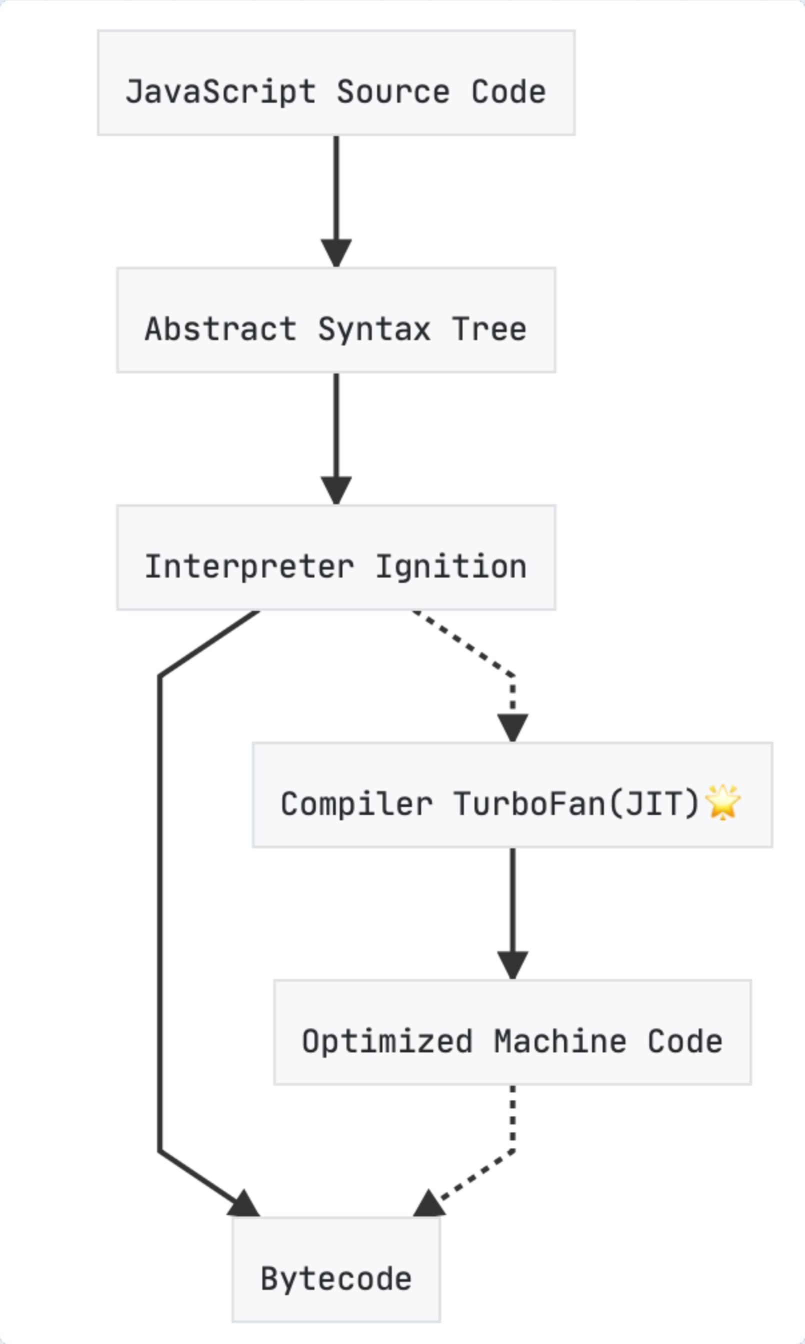
```
> a=1
<· 1
>
```

JavaScript Engine

How to Test JavaScript Engine?

JavaScript Engine

- input: xxx.js
    - 接受结构化输入
- xxx.js 👉 AST 👉 Bytecode
    - 执行Bytecode



JavaScript Source Code

Abstract Syntax Tree

Interpreter Ignition

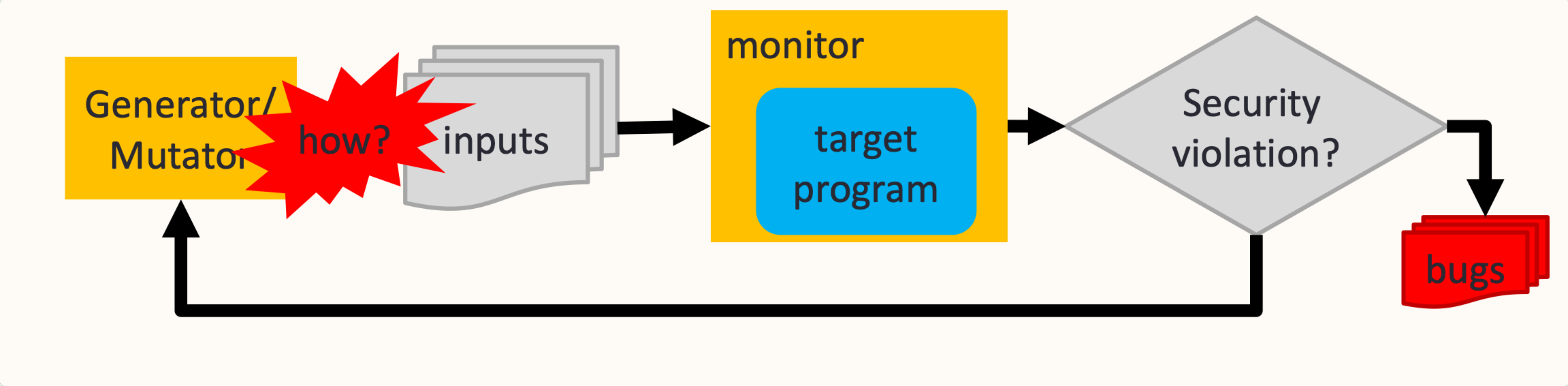
Compiler TurboFan(JIT)🌟

Optimized Machine Code

Bytecode

Fuzzing：A Simple Fuzz Testing Framework

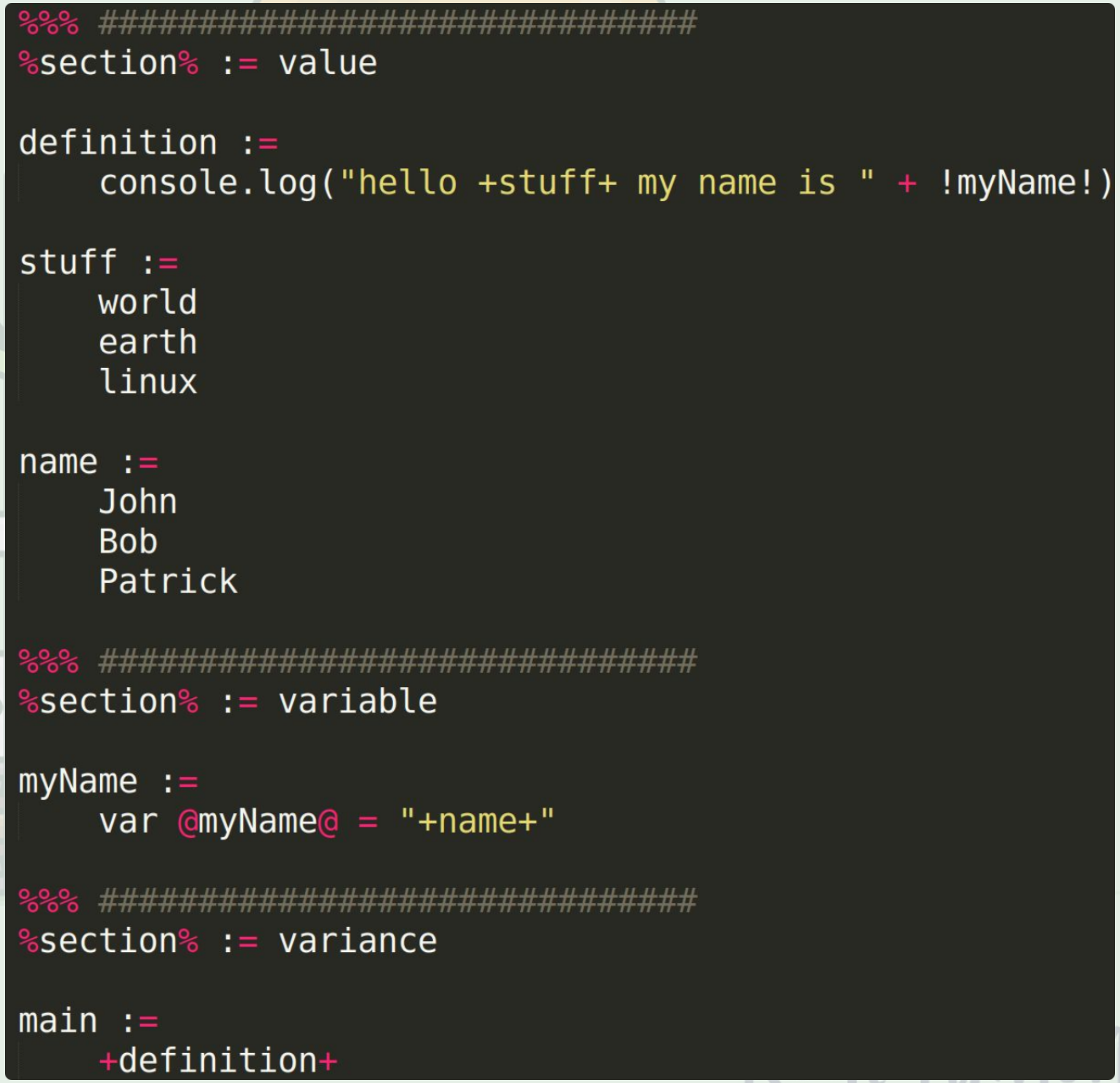
- input: 一系列xxx.js作为testcases
- monitor: 监控是否有testcases使得JavaScript Engine运行后异常退出
- 筛选出bug



Grammar-based Fuzzing：Generate testcases

Generation-based fuzzer——dharma

- context-free grammar
- 人工编写语法规则、terminal等
- 生成js样例

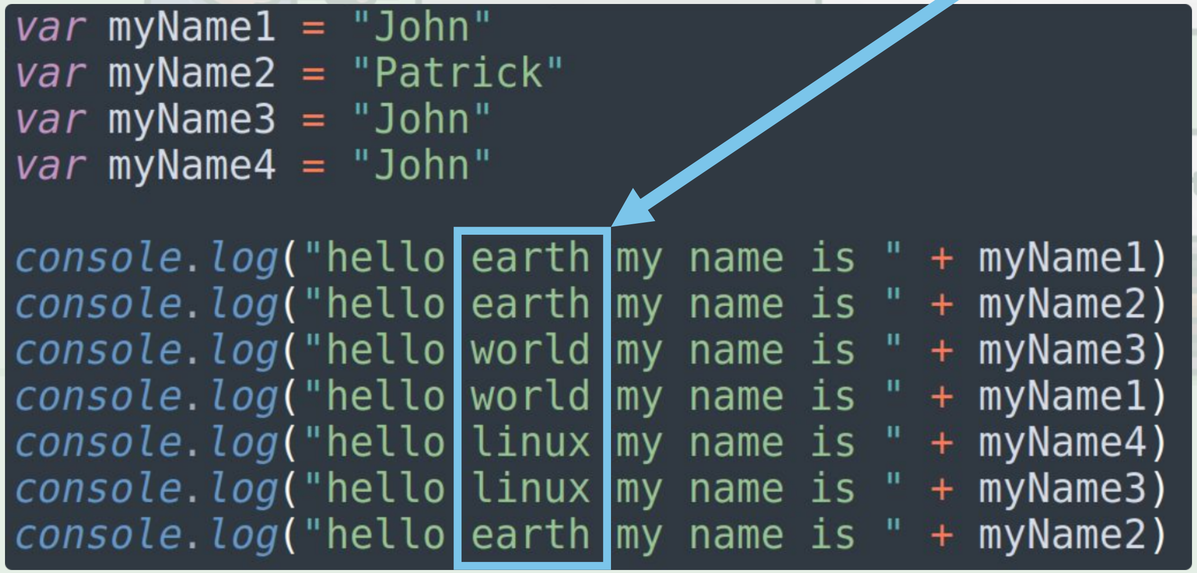


```
%%% ##############################
%section% := value

definition :=
    console.log("hello +stuff+ my name is " + !myName!)

stuff :=
    world
    earth
    linux

name :=
    John
    Bob
    Patrick

%%% ##############################
%section% := variable

myName :=
    var @myName@ = "+name+"

%%% ##############################
%section% := variance

main :=
    +definition+
```

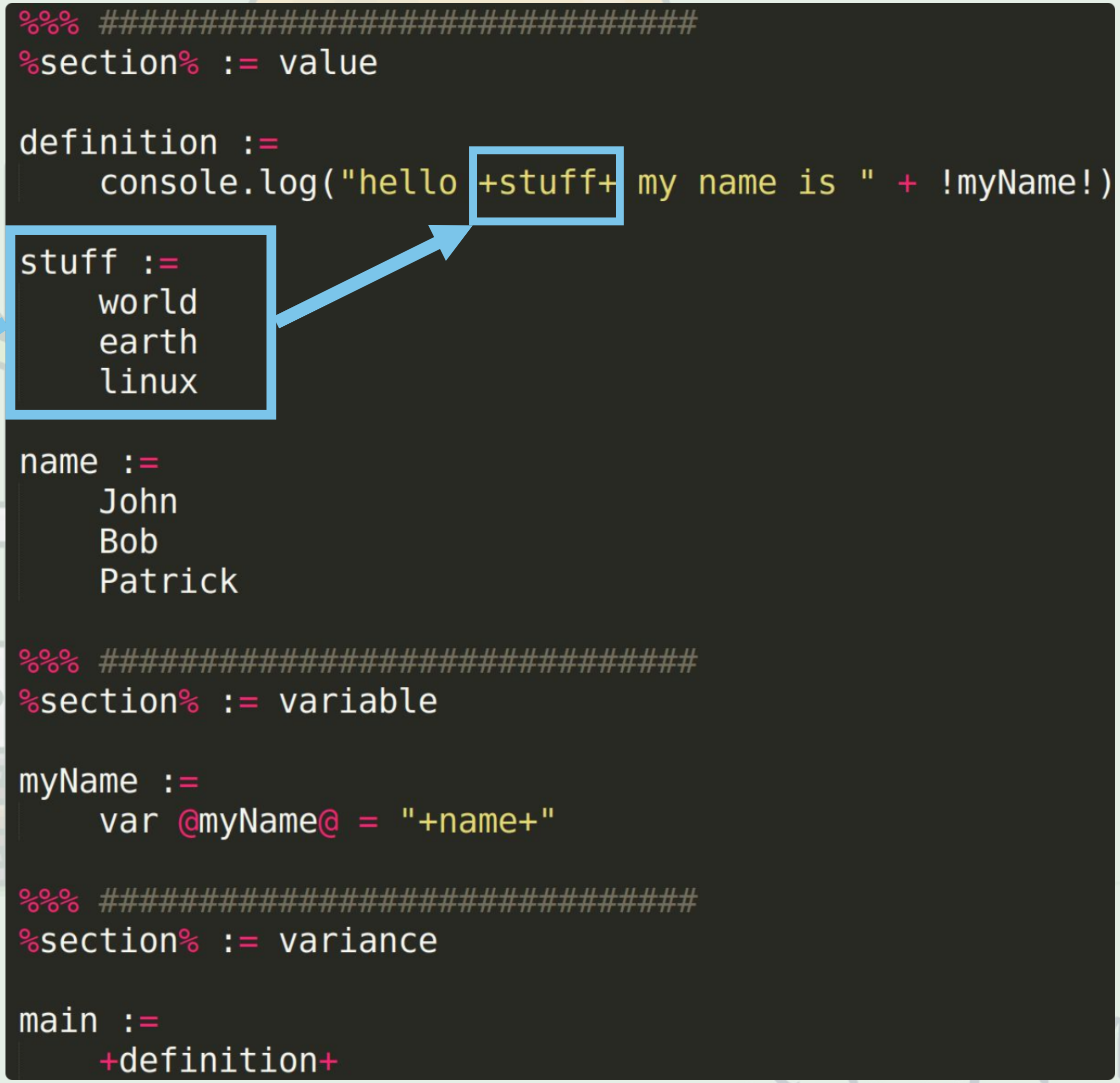
Grammar-based Fuzzing：Generate testcases

Generation-based fuzzer——dharma

- context-free grammar
- 人工编写语法规则、terminal等
- 生成js样例



```
var myName1 = "John"
var myName2 = "Patrick"
var myName3 = "John"
var myName4 = "John"

console.log("hello earth my name is " + myName1)
console.log("hello earth my name is " + myName2)
console.log("hello world my name is " + myName3)
console.log("hello world my name is " + myName1)
console.log("hello linux my name is " + myName4)
console.log("hello linux my name is " + myName3)
console.log("hello earth my name is " + myName2)
```

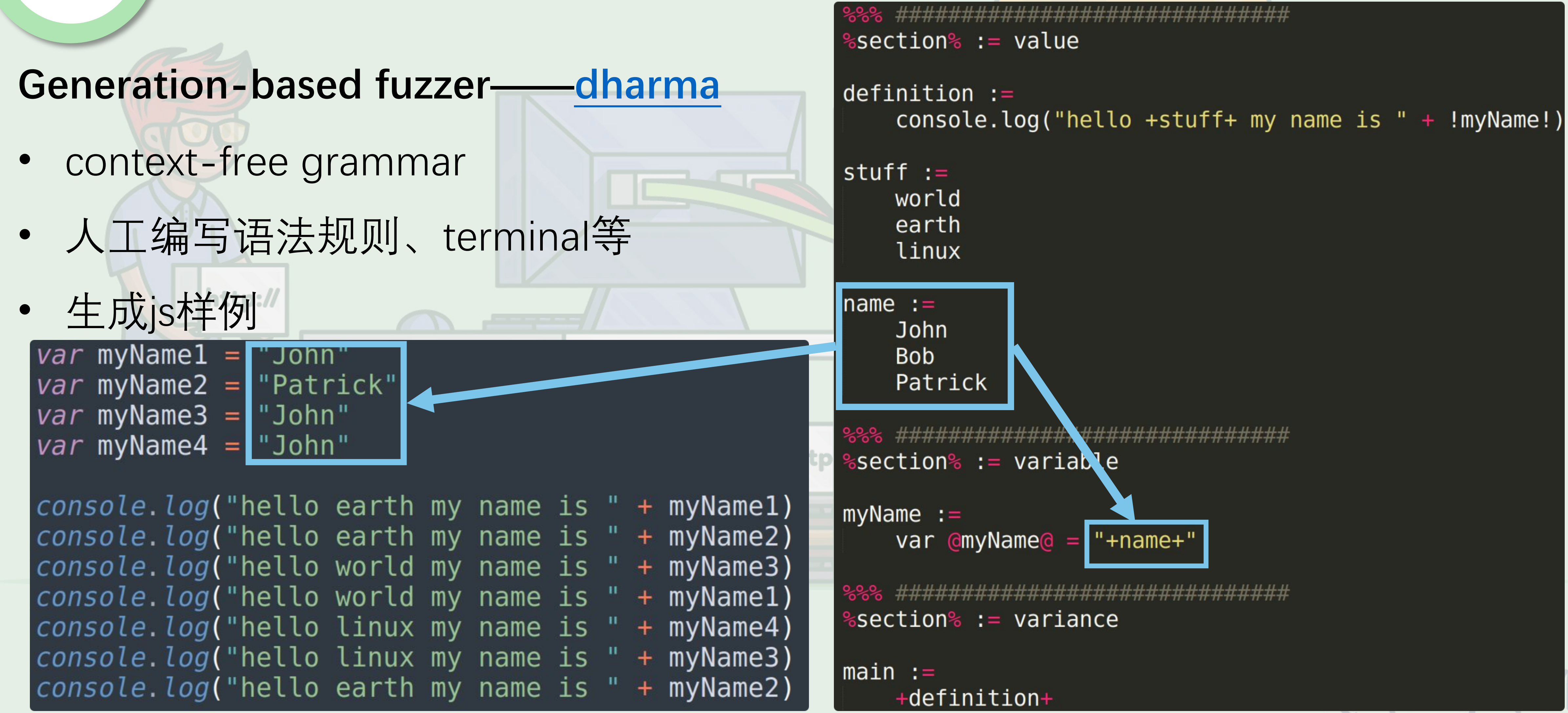


```
%%% ##############################
%section% := value

definition :=
    console.log("hello +stuff+ my name is " + !myName!)

stuff :=
    world
    earth
    linux

name :=
    John
    Bob
    Patrick

%%% ##############################
%section% := variable

myName :=
    var @myName@ = "+name+"

%%% ##############################
%section% := variance

main :=
    +definition+
```

Grammar-based Fuzzing：Generate testcases

Generation-based fuzzer——dharma

- context-free grammar
- 人工编写语法规则、terminal等
- 生成js样例



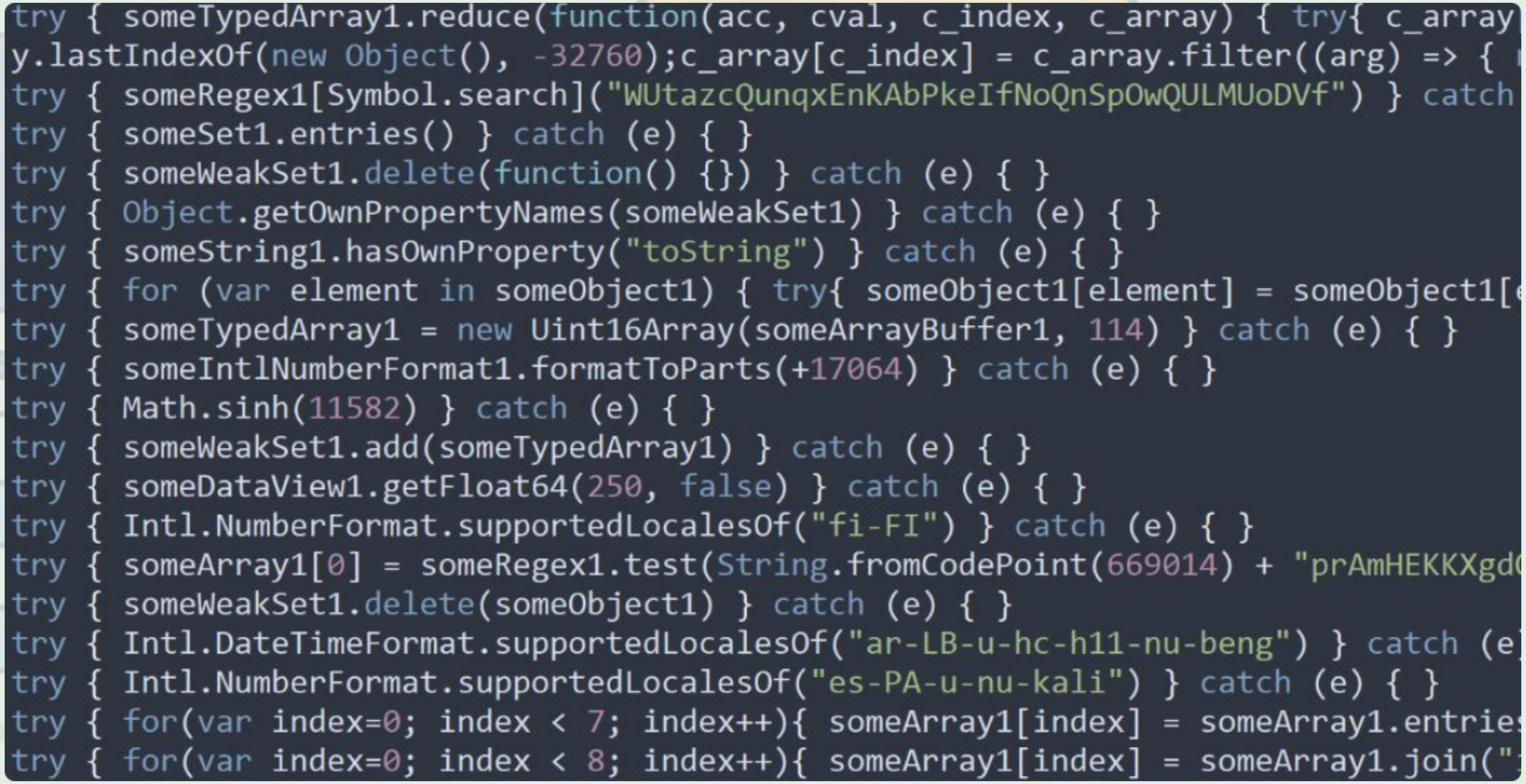
```
var myName1 = "John"
var myName2 = "Patrick"
var myName3 = "John"
var myName4 = "John"

console.log("hello earth my name is " + myName1)
console.log("hello earth my name is " + myName2)
console.log("hello world my name is " + myName3)
console.log("hello world my name is " + myName1)
console.log("hello linux my name is " + myName4)
console.log("hello linux my name is " + myName3)
console.log("hello earth my name is " + myName2)
```

```
%%% ##############################
%section% := value

definition :=
    console.log("hello +stuff+ my name is " + !myName!)

stuff :=
    world
    earth
    linux

name :=
    John
    Bob
    Patrick

%%% ##############################
%section% := variable

myName :=
    var @myName@ = "+name+"

%%% ##############################
%section% := variance

main :=
    +definition+
```

Grammar-based Fuzzing：Generate testcases

Domato

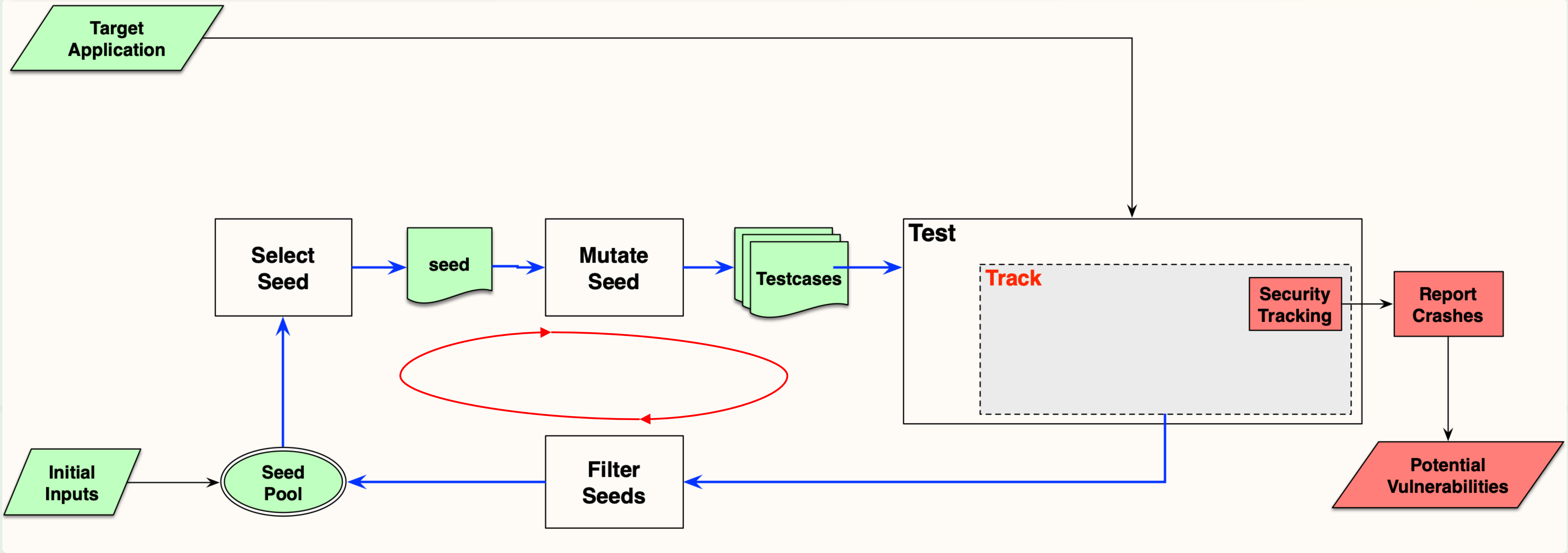
- 测试API
  - 参数类型、数量随机
  - Try-catch
- 缺点🤔
  - 语义正确率低
  - 人工编写语法模版



```
try { someTypedArray1.reduce(function(acc, cval, c_index, c_array) { try{ c_array
y.lastIndexOf(new Object(), -32760);c_array[c_index] = c_array.filter((arg) => {
try { someRegex1[Symbol.search]("WUtazcQunqxEnKAbPkeIfNoQnSpOwQULMUoDVf") } catch
try { someSet1.entries() } catch (e) { }
try { someWeakSet1.delete(function() {}) } catch (e) { }
try { Object.getOwnPropertyNames(someWeakSet1) } catch (e) { }
try { someString1.hasOwnProperty("toString") } catch (e) { }
try { for (var element in someObject1) { try{ someObject1[element] = someObject1[
try { someTypedArray1 = new Uint16Array(someArrayBuffer1, 114) } catch (e) { }
try { someIntlNumberFormat1.formatToParts(+17064) } catch (e) { }
try { Math.sinh(11582) } catch (e) { }
try { someWeakSet1.add(someTypedArray1) } catch (e) { }
try { someDataView1.getFloat64(250, false) } catch (e) { }
try { Intl.NumberFormat.supportedLocalesOf("fi-FI") } catch (e) { }
try { someArray1[0] = someRegex1.test(String.fromCodePoint(669014) + "prAmHEKKXgd
try { someWeakSet1.delete(someObject1) } catch (e) { }
try { Intl.DateTimeFormat.supportedLocalesOf("ar-LB-u-hc-h11-nu-beng") } catch (e
try { Intl.NumberFormat.supportedLocalesOf("es-PA-u-nu-kali") } catch (e) { }
try { for(var index=0; index < 7; index++){ someArray1[index] = someArray1.entrie
try { for(var index=0; index < 8; index++){ someArray1[index] = someArray1.join("
```

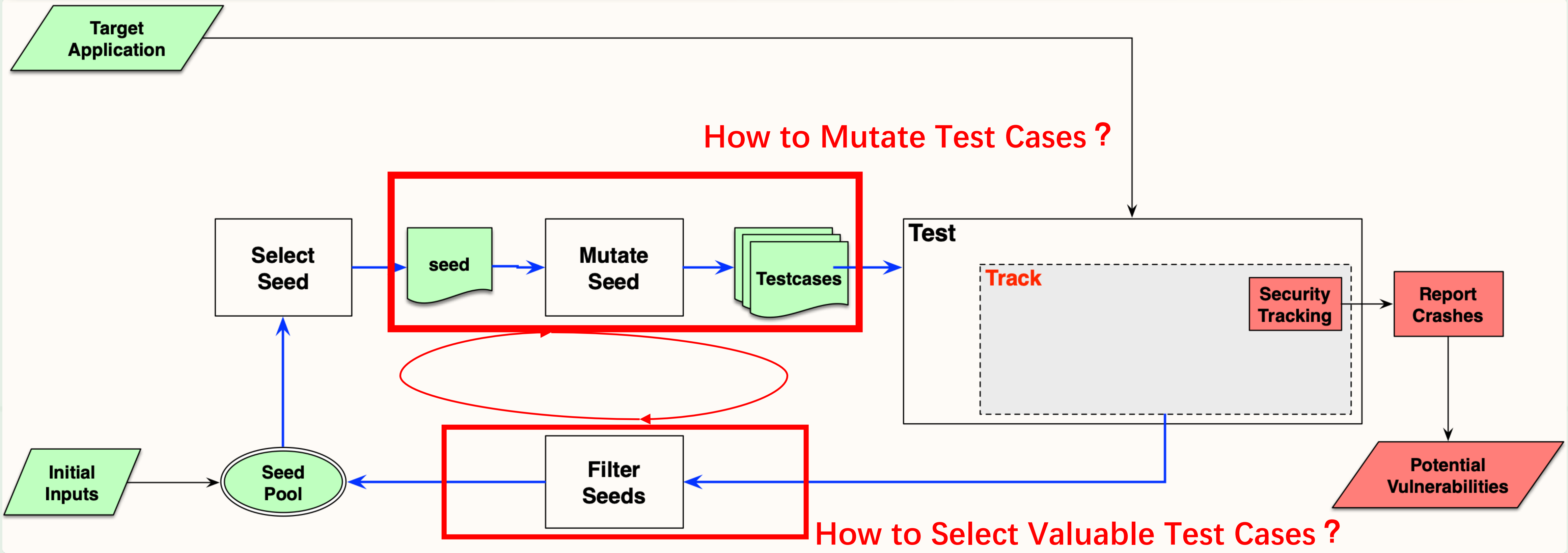
AFL：Mutate testcases

1. 选取一批testcases作为初始seeds
2. 变异生成更多的testcases
3. 保留一些testcases等待下一轮变异



AFL：Mutate testcases

1. 选取一批testcases作为初始seeds
2. 变异生成更多的testcases
3. 保留一些testcases等待下一轮变异



AFL：Mutate Strategy

Mutate Strategy（for libxxx)

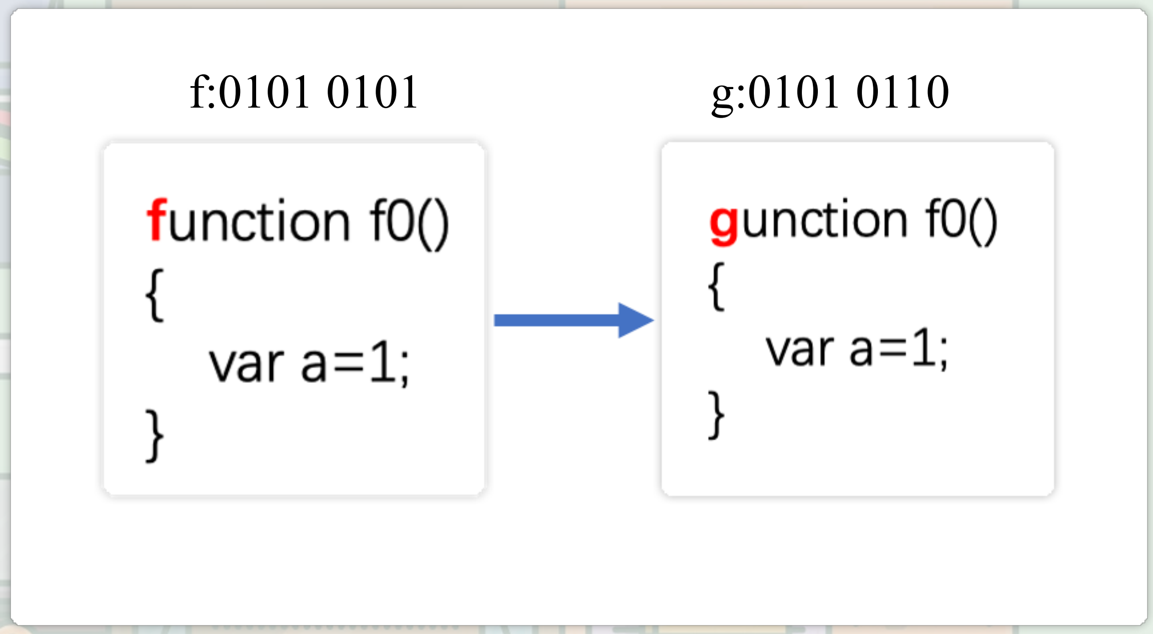
① bitflip

② interest

③ arithmetic

④ Dictionary

⑤ havoc



f:0101 0101

```
function f0()
{
    var a=1;
}
```

g:0101 0110

```
gunction f0()
{
    var a=1;
}
```

bit级别变异产生大量无效testcases

AFL：Mutate Strategy

Mutate Strategy（for libxxx）

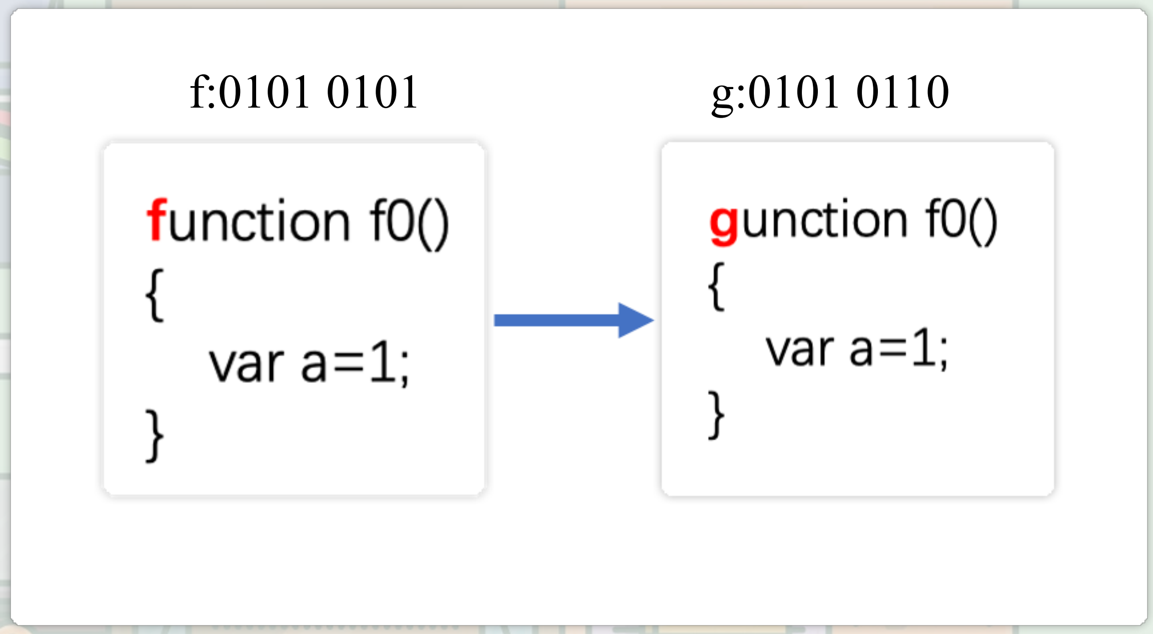
① bitflip

② interest

③ arithmetic

④ Dictionary

⑤ havoc



f:0101 0101

```
function f0()
{
    var a=1;
}
```

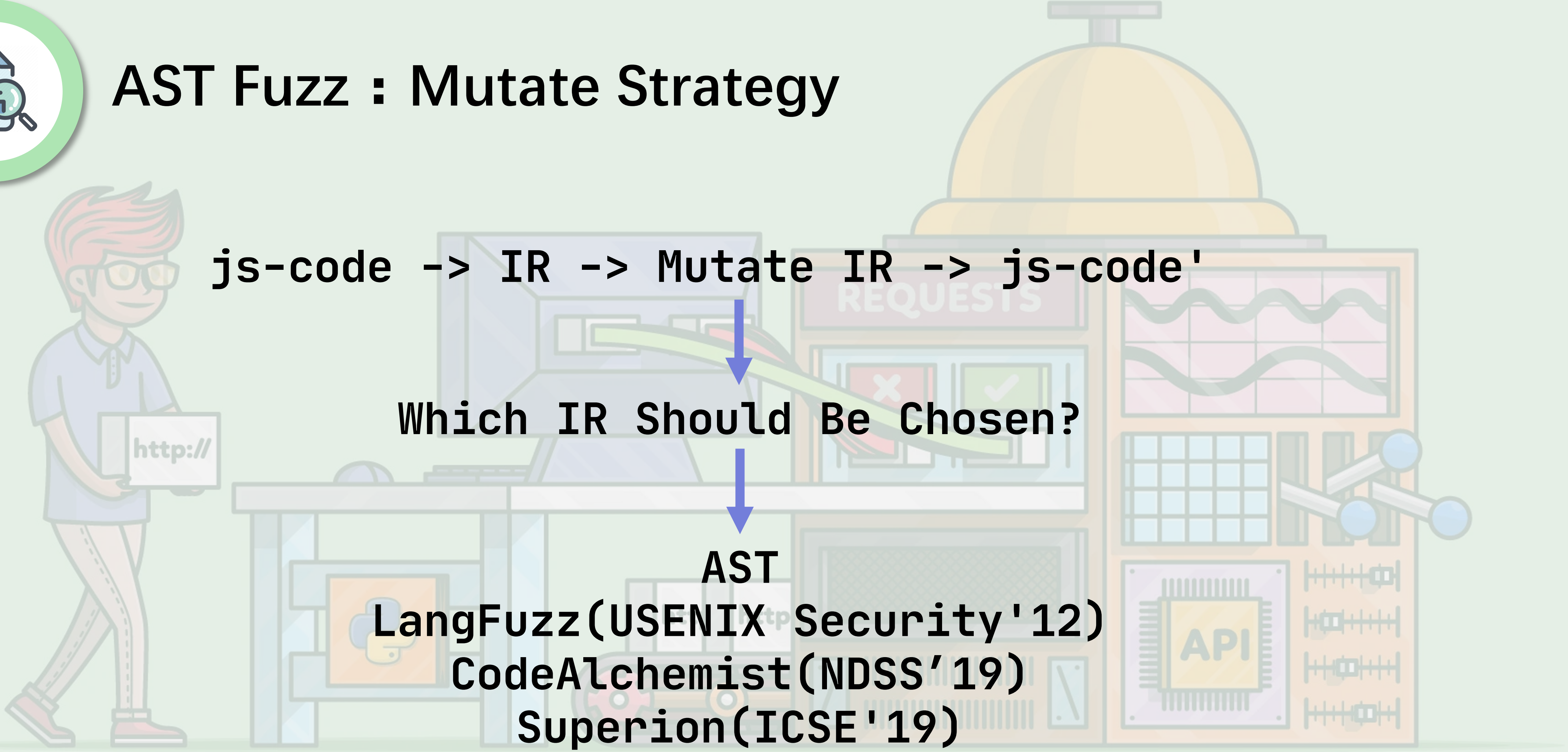
g:0101 0110

```
gunction f0()
{
    var a=1;
}
```

bit级别变异产生大量无效testcases

How to improve Improve the Effectiveness of Mutated JS Testcases？

AST Fuzz : Mutate Strategy



js-code -> IR -> Mutate IR -> js-code'

↓

Which IR Should Be Chosen?

↓

AST

LangFuzz(USENIX Security'12)

CodeAlchemist(NDSS’19)

Superion(ICSE'19)

DIE(S&P'20)

…

AST Fuzz : Mutate Strategy

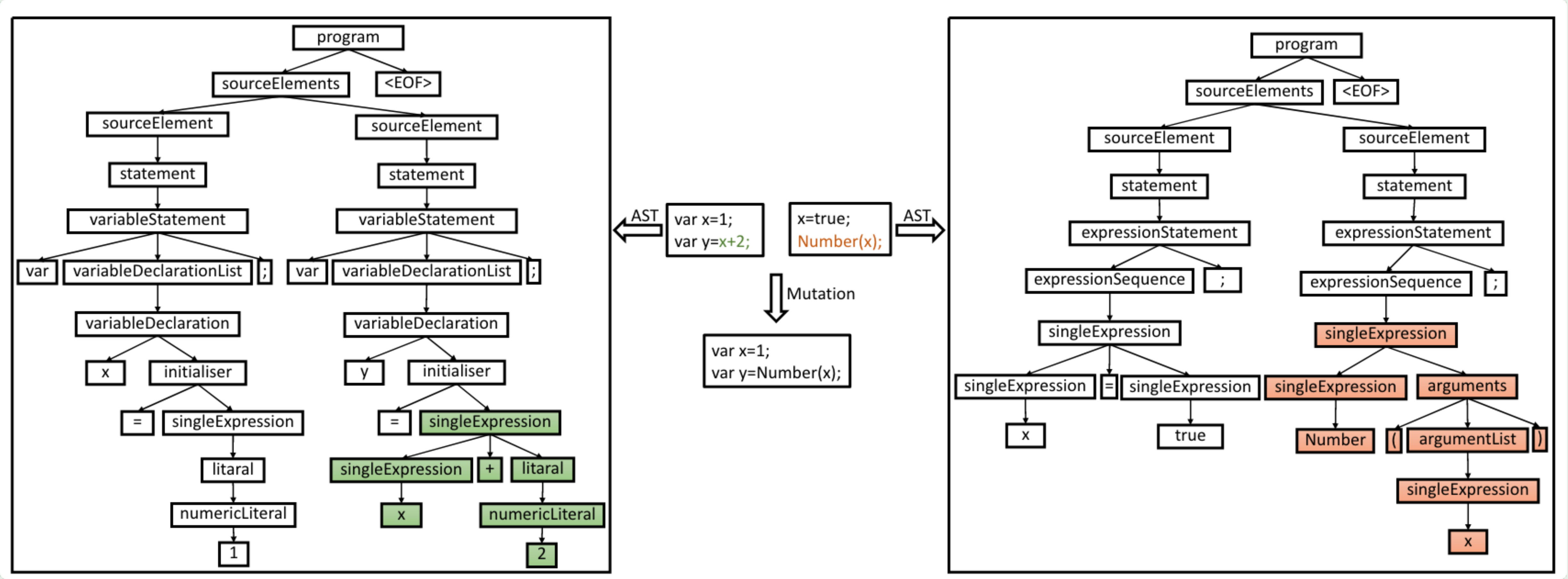
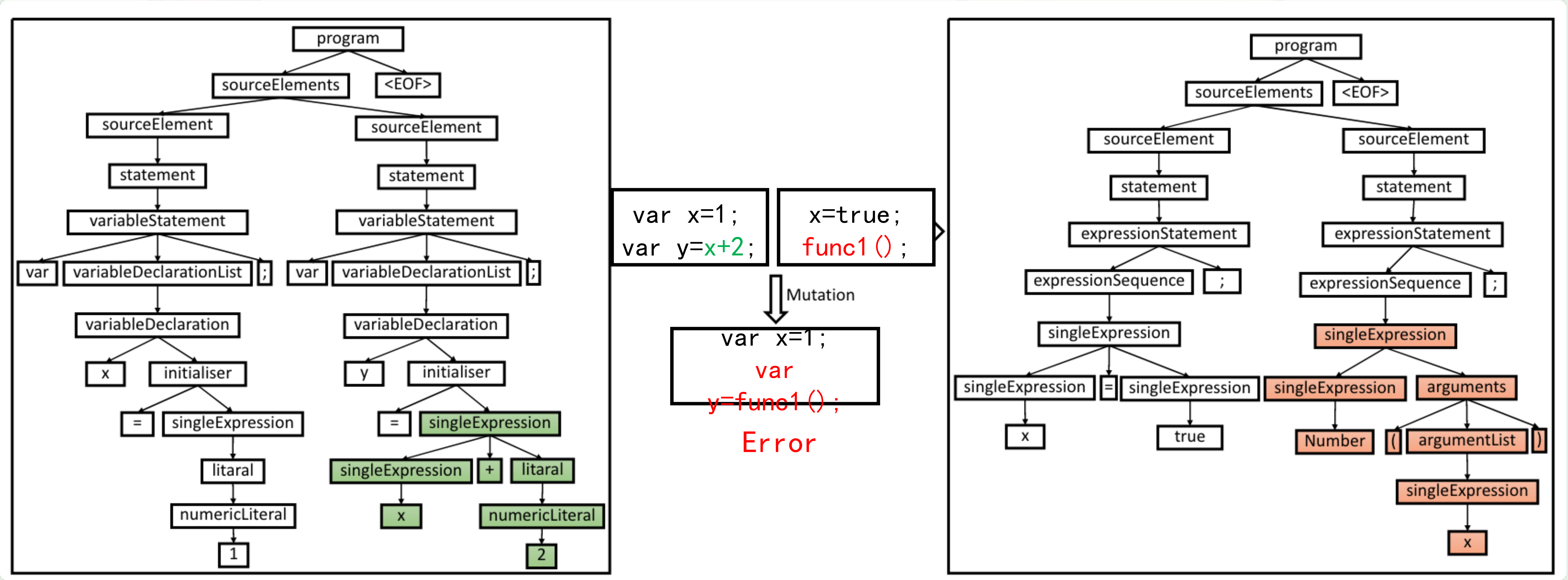


Figure from Superion (ICSE'19)

AST Fuzz : Mutate Strategy

Undefined Behavior？



```
var x=1;
var y=x+2;
```

```
x=true;
func1();
```

Mutation

```
var x=1;
var
y=func1();
```

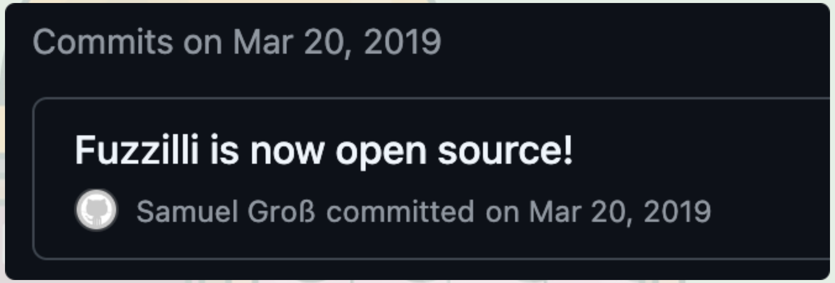
Error

Figure from Superion (ICSE'19)

IR Fuzz： improve the semantic correctness

Fuzzilli (NDSS'23)

- 自定义IR——FuzzIL(Groß's Masters Thesis 2018)



Commits on Mar 20, 2019

Fuzzilli is now open source!

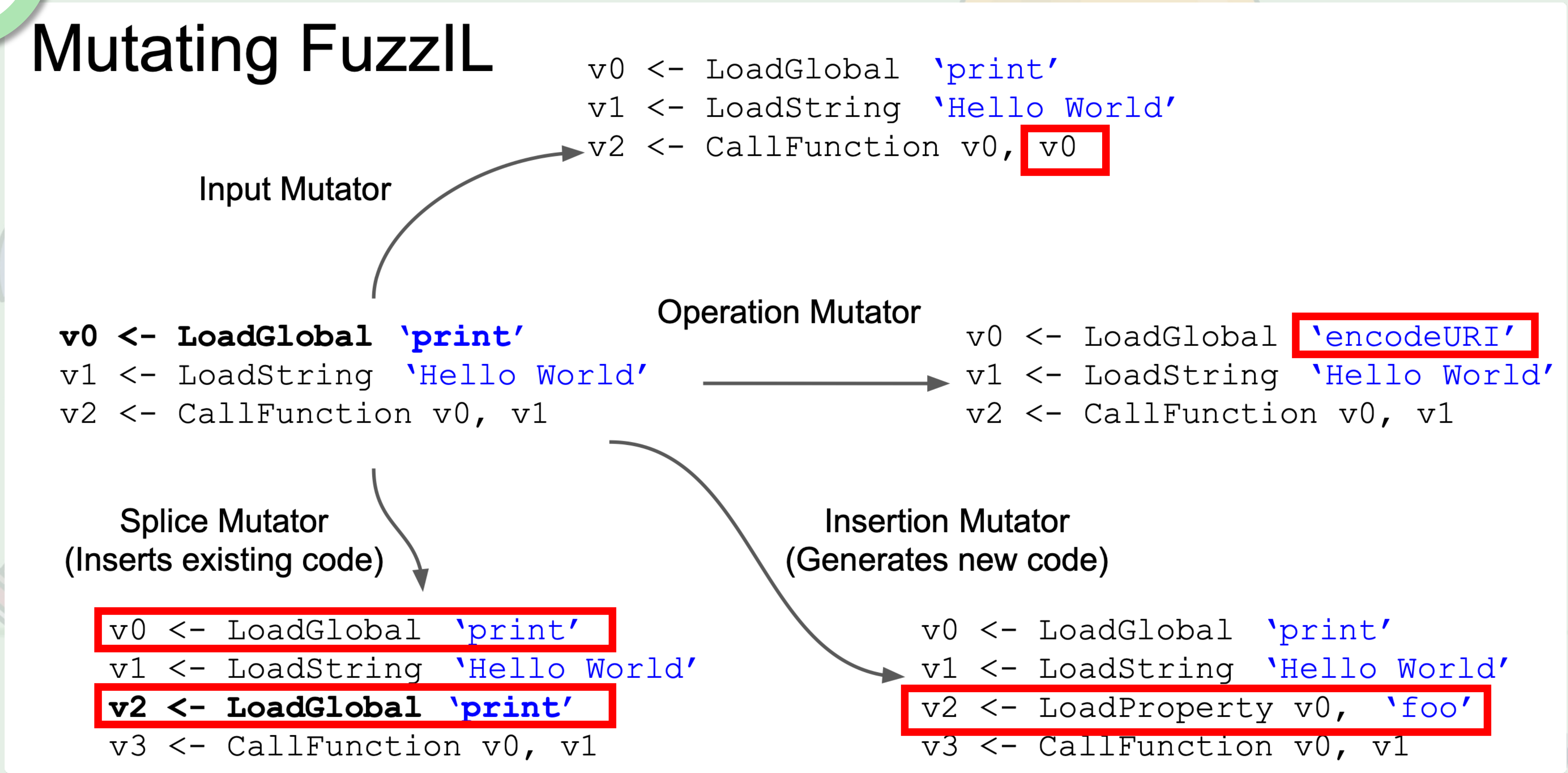
Samuel Groß committed on Mar 20, 2019

```
; Example FuzzIL program
v0 <- LoadInt '0'
v1 <- LoadInt '10'
v2 <- LoadInt '1'
v3 <- Phi v0
BeginFor v0, '<', v1, '+', v2 -> v4
   v6 <- BinaryOperation v3, '+', v4
   Copy v3, v6
EndFor
v7 <- LoadString 'Result: '
v8 <- BinaryOperation v7, '+', v3
v9 <- LoadGlobal 'console'
v10 <- CallMethod v9, 'log', [v8]
```

Lift →

```
// Trivial lifting
const v0 = 0;
const v1 = 10;
const v2 = 1;
let v3 = v0;
for (let v4 = v0; v4 < v1; v4 = v4 + v2) {
    const v6 = v3 + v4;
    v3 = v6;
}
const v7 = ”Result:”;
const v8 = v7 + v3;
const v9 = console;
const v10 = v9.log(v8);
```

Fuzzilli： Mutate FuzzIL



Mutating FuzzIL

```
v0 <- LoadGlobal 'print'
v1 <- LoadString 'Hello World'
v2 <- CallFunction v0, v1
```

Input Mutator

```
v0 <- LoadGlobal 'print'
v1 <- LoadString 'Hello World'
v2 <- CallFunction v0, v0
```

Operation Mutator

```
v0 <- LoadGlobal 'encodeURI'
v1 <- LoadString 'Hello World'
v2 <- CallFunction v0, v1
```

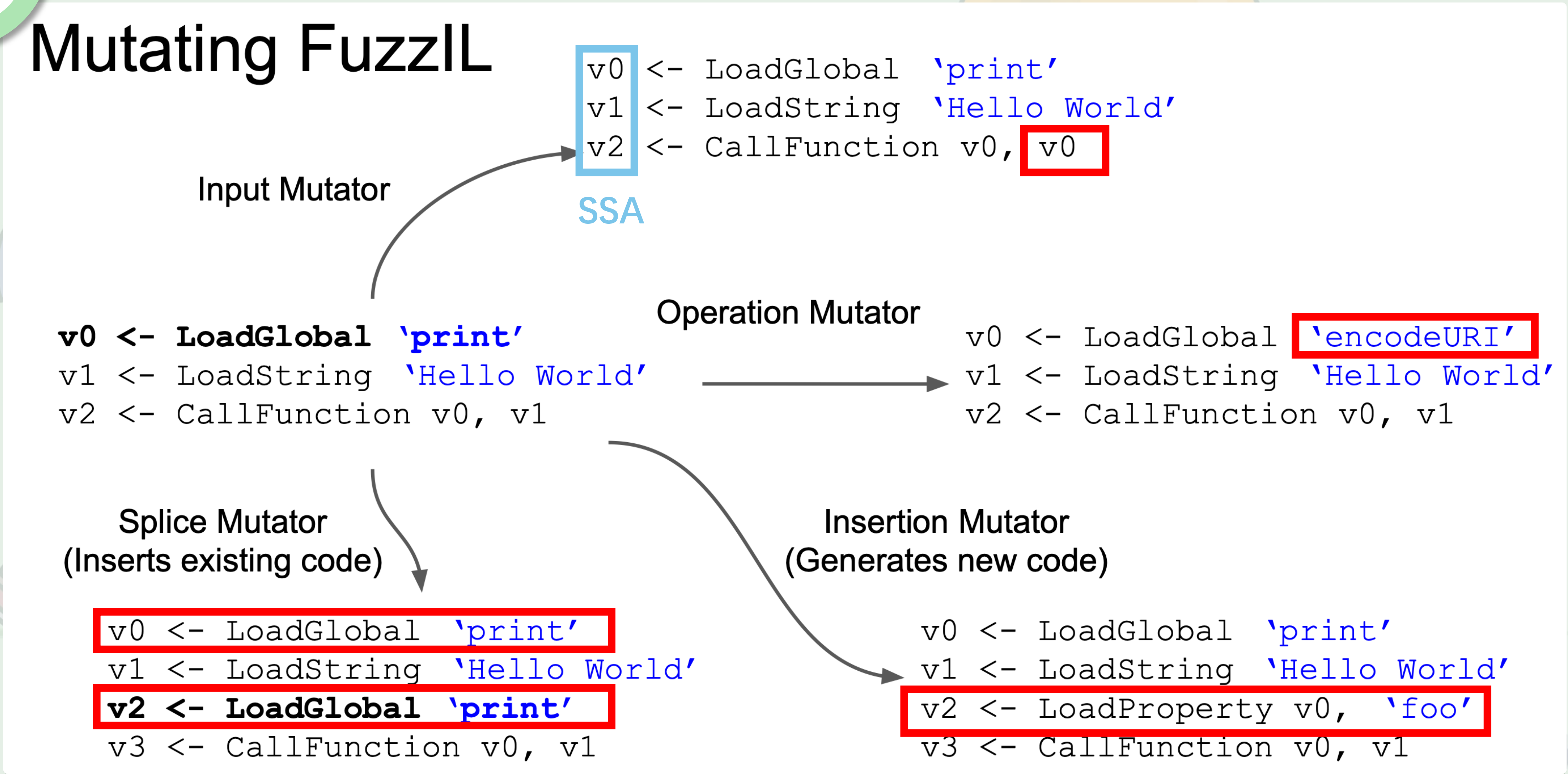
Splice Mutator
(Inserts existing code)

```
v0 <- LoadGlobal 'print'
v1 <- LoadString 'Hello World'
v2 <- LoadGlobal 'print'
v3 <- CallFunction v0, v1
```

Insertion Mutator
(Generates new code)

```
v0 <- LoadGlobal 'print'
v1 <- LoadString 'Hello World'
v2 <- LoadProperty v0, 'foo'
v3 <- CallFunction v0, v1
```

Fuzzilli： Mutate FuzzIL



Mutating FuzzIL

```
v0 <- LoadGlobal  'print'
v1 <- LoadString  'Hello World'
v2 <- CallFunction v0, v1
```

Input Mutator

```
v0 <- LoadGlobal  'print'
v1 <- LoadString  'Hello World'
v2 <- CallFunction v0, v0
```

SSA

Operation Mutator

```
v0 <- LoadGlobal  'encodeURI'
v1 <- LoadString  'Hello World'
v2 <- CallFunction v0, v1
```

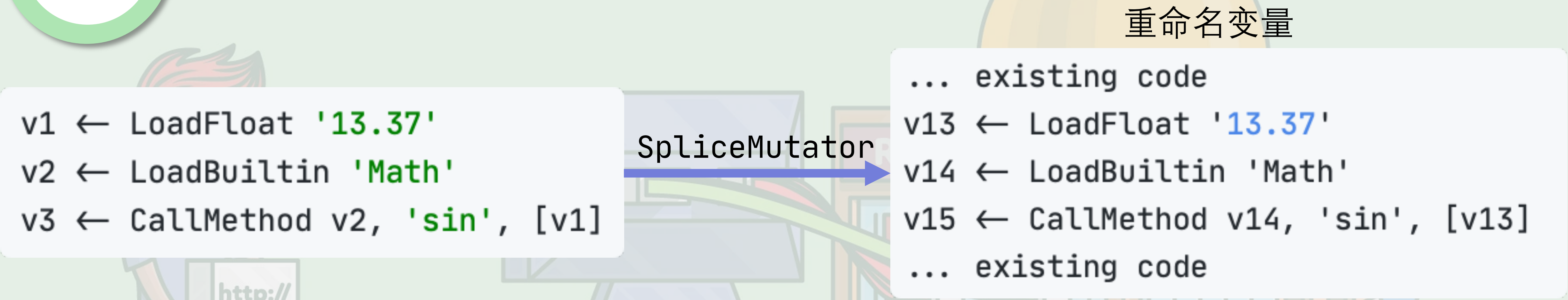
Splice Mutator
(Inserts existing code)

```
v0 <- LoadGlobal  'print'
v1 <- LoadString  'Hello World'
v2 <- LoadGlobal  'print'
v3 <- CallFunction v0, v1
```

Insertion Mutator
(Generates new code)

```
v0 <- LoadGlobal  'print'
v1 <- LoadString  'Hello World'
v2 <- LoadProperty v0,  'foo'
v3 <- CallFunction v0, v1
```

Fuzzilli： Analyze



```
v1 ← LoadFloat '13.37'
v2 ← LoadBuiltin 'Math'
v3 ← CallMethod v2, 'sin', [v1]
```

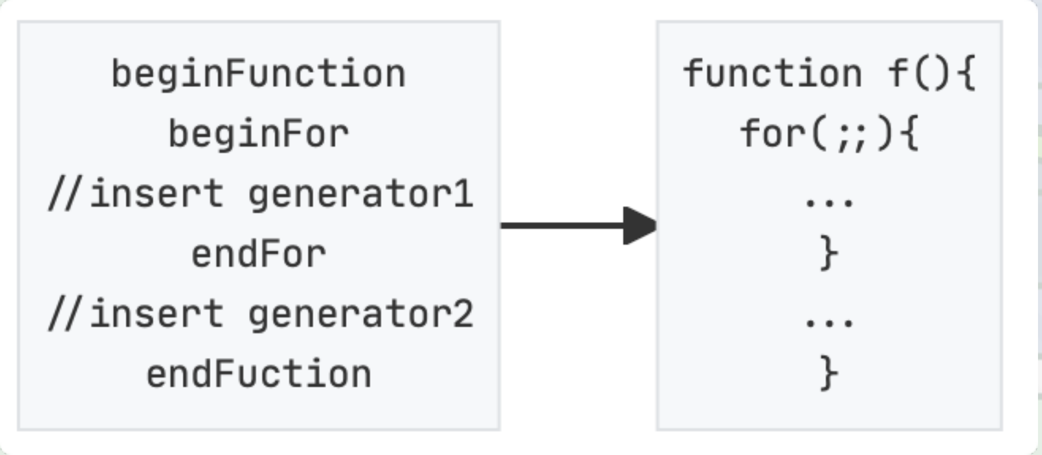
SpliceMutator

重命名变量

```
... existing code
v13 ← LoadFloat '13.37'
v14 ← LoadBuiltin 'Math'
v15 ← CallMethod v14, 'sin', [v13]
... existing code
```

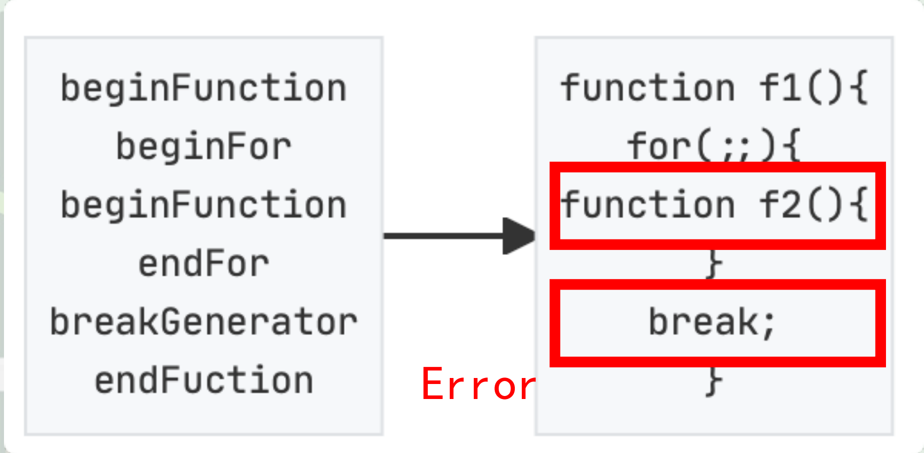
- Context Analyze
- Scope Analyze
- Type Analyze

Fuzzilli： Context Analyze



```
beginFunction
beginFor
//insert generator1
endFor
//insert generator2
endFuction
```

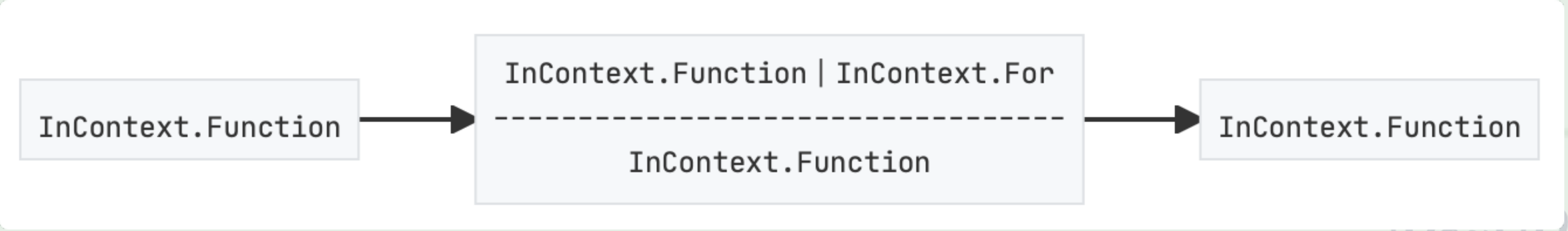
```
function f(){
for(;;){
...
}
...
}
```



```
beginFunction
beginFor
beginFunction
endFor
breakGenerator
endFuction
```

Error

```
function f1(){
for(;;){
function f2(){
}
break;
}
```



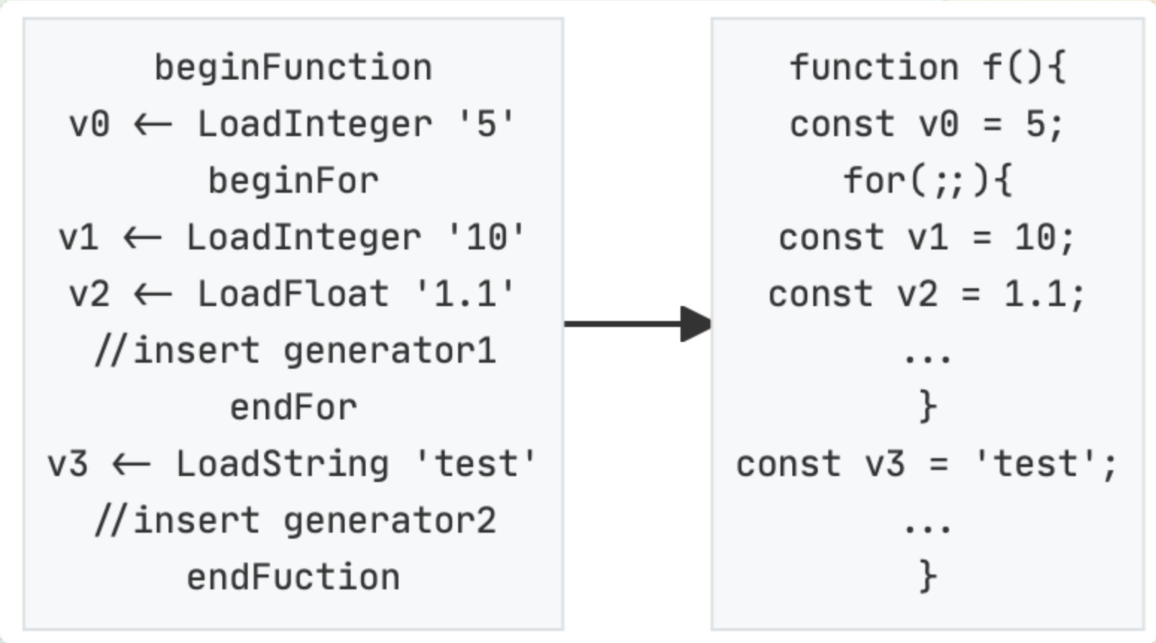
```
InContext.Function
```

```
InContext.Function | InContext.For
-----------------------------------
InContext.Function
```

```
InContext.Function
```

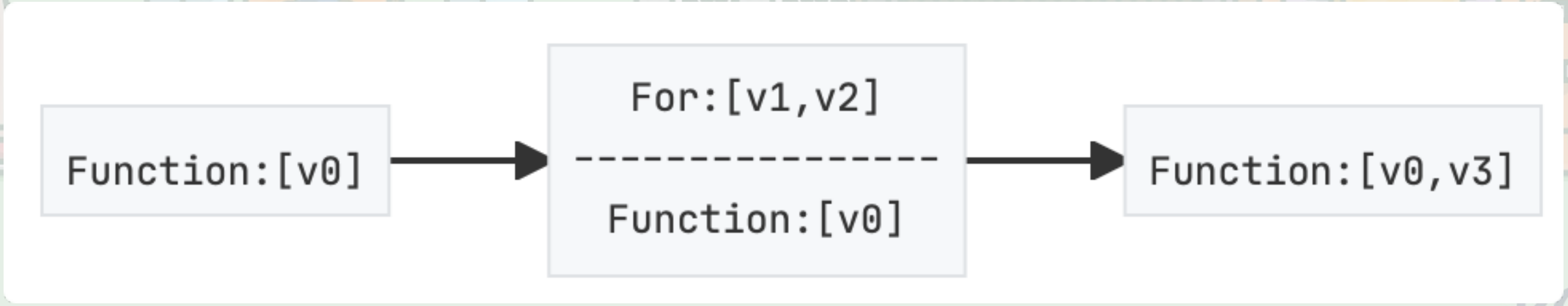
栈维护上下文信息

Fuzzilli： Scope Analyze



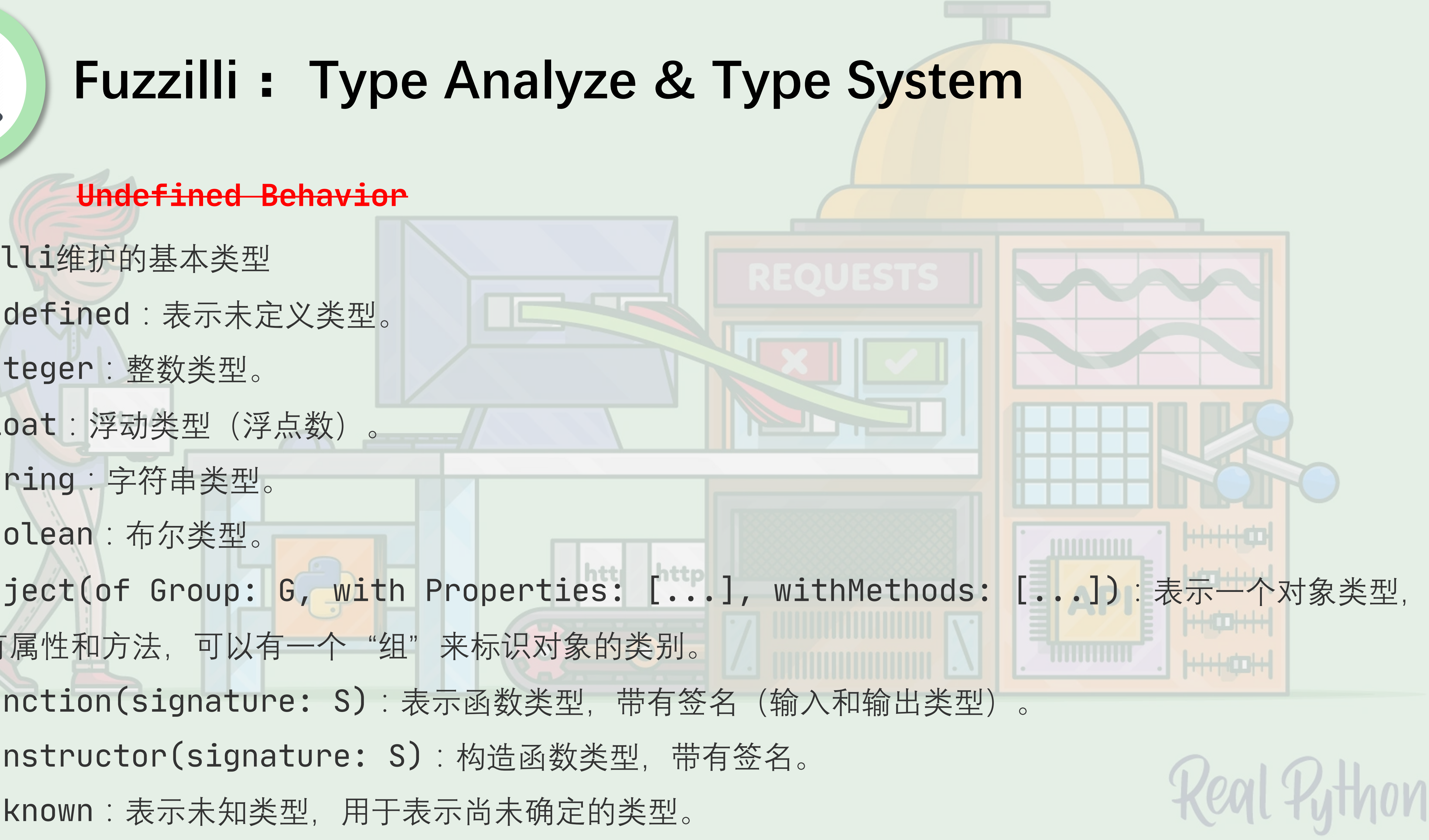
```
beginFunction
v0 ← LoadInteger '5'
beginFor
v1 ← LoadInteger '10'
v2 ← LoadFloat '1.1'
//insert generator1
endFor
v3 ← LoadString 'test'
//insert generator2
endFuction
```

```
function f(){
const v0 = 5;
for(;;){
const v1 = 10;
const v2 = 1.1;
...
}
const v3 = 'test';
...
}
```



栈维护变量定义信息

Fuzzilli： Type Analyze & Type System

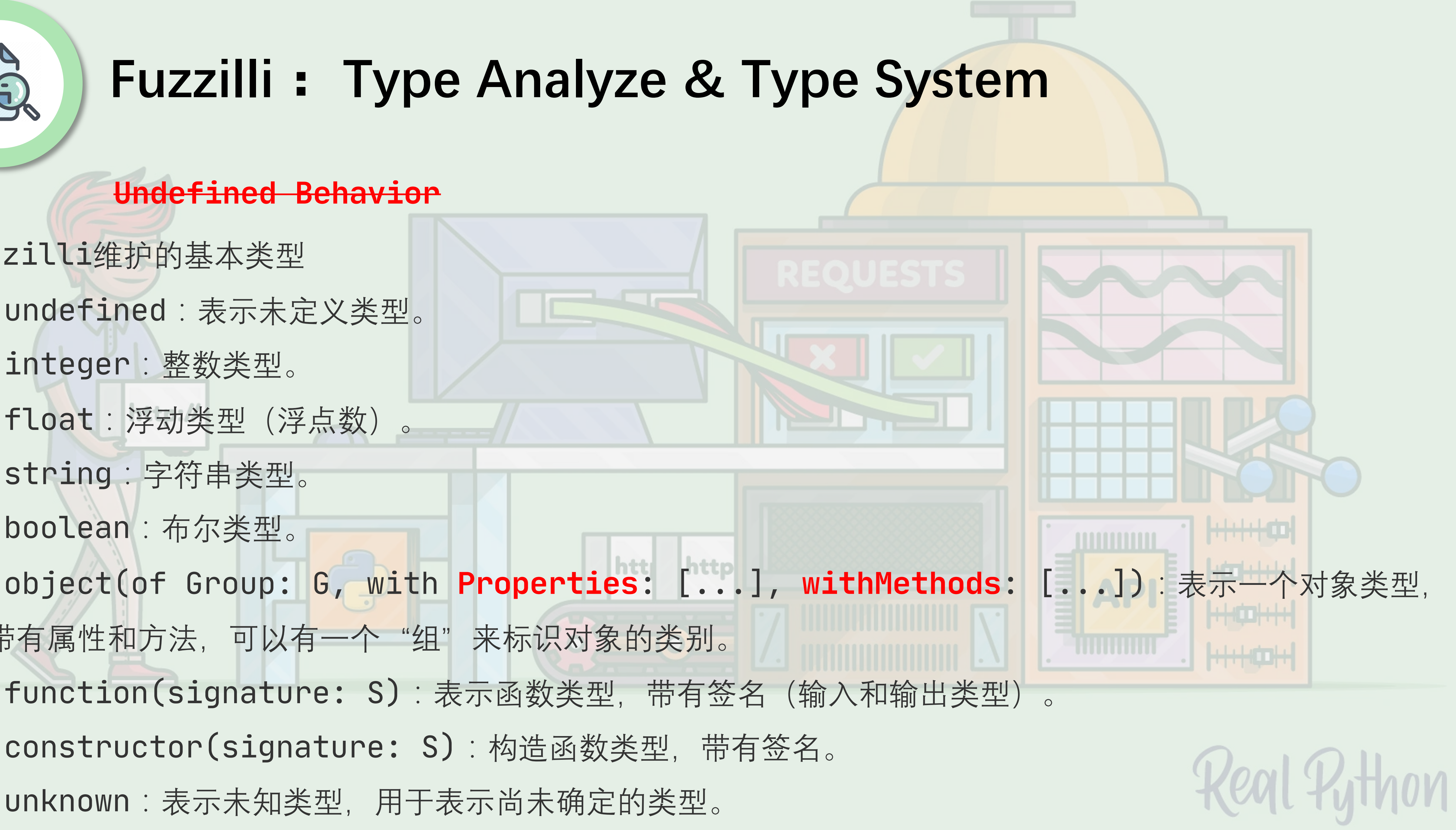


~~Undefined Behavior~~

Fuzzilli维护的基本类型

- .undefined：表示未定义类型。
- .integer：整数类型。
- .float：浮动类型（浮点数）。
- .string：字符串类型。
- .boolean：布尔类型。
- .object(of Group: G, with Properties: [...], withMethods: [...])：表示一个对象类型，带有属性和方法，可以有一个“组”来标识对象的类别。
- .function(signature: S)：表示函数类型，带有签名（输入和输出类型）。
- .constructor(signature: S)：构造函数类型，带有签名。
- .unknown：表示未知类型，用于表示尚未确定的类型。

Fuzzilli： Type Analyze & Type System



~~Undefined Behavior~~

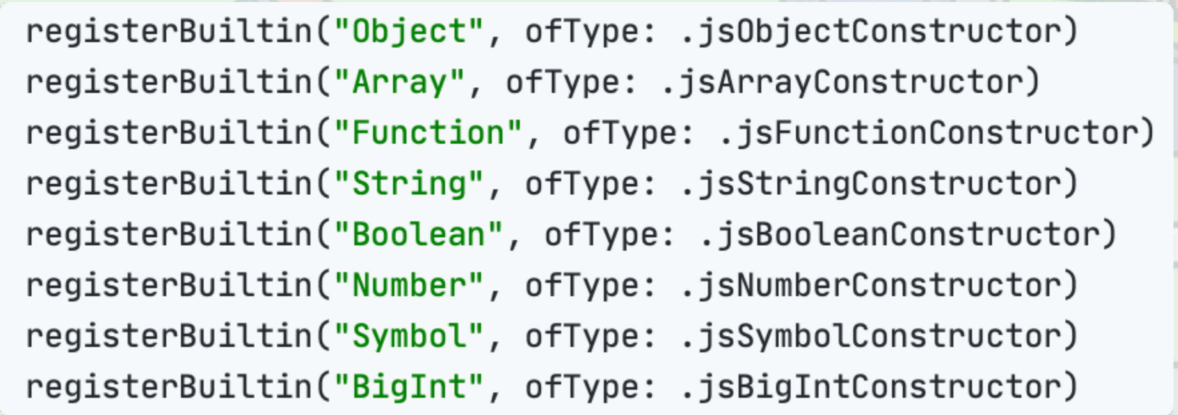
Fuzzilli维护的基本类型

- .undefined：表示未定义类型。
- .integer：整数类型。
- .float：浮动类型（浮点数）。
- .string：字符串类型。
- .boolean：布尔类型。
- .object(of Group: G, with **Properties**: [...], **withMethods**: [...])：表示一个对象类型，带有属性和方法，可以有一个“组”来标识对象的类别。
- .function(signature: S)：表示函数类型，带有签名（输入和输出类型）。
- .constructor(signature: S)：构造函数类型，带有签名。
- .unknown：表示未知类型，用于表示尚未确定的类型。

Fuzzilli： Type Analyze & Type System

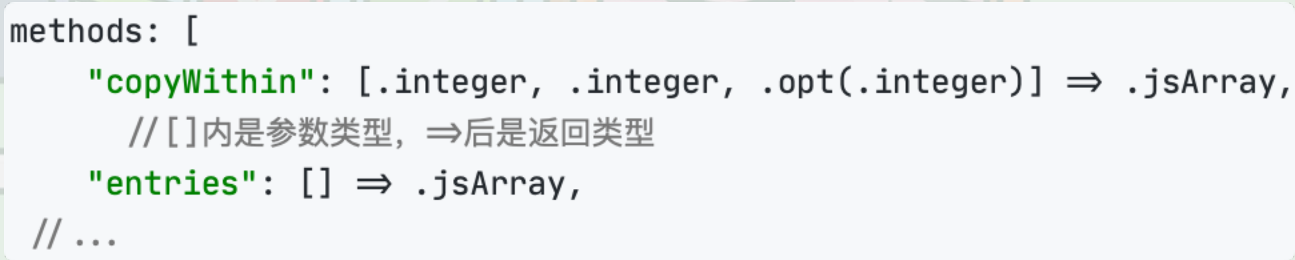
Fuzzilli在函数调用方面做的约束——初版类型系统（Commit 98e605e）

增加Built-in(内建函数)



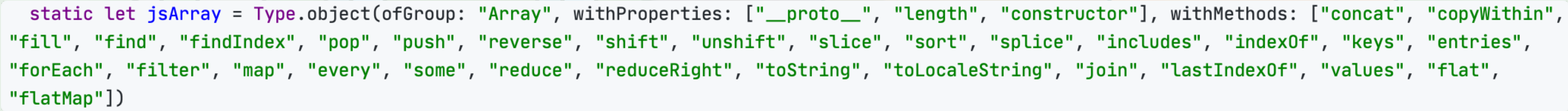
```
registerBuiltin("Object", ofType: .jsObjectConstructor)
registerBuiltin("Array", ofType: .jsArrayConstructor)
registerBuiltin("Function", ofType: .jsFunctionConstructor)
registerBuiltin("String", ofType: .jsStringConstructor)
registerBuiltin("Boolean", ofType: .jsBooleanConstructor)
registerBuiltin("Number", ofType: .jsNumberConstructor)
registerBuiltin("Symbol", ofType: .jsSymbolConstructor)
registerBuiltin("BigInt", ofType: .jsBigIntConstructor)
```

约束API调用传入参数类型与返回类型



```
methods: [
    "copyWithin": [.integer, .integer, .opt(.integer)] => .jsArray,
      //[]内是参数类型，=>后是返回类型
    "entries": [] => .jsArray,
 // ...
```

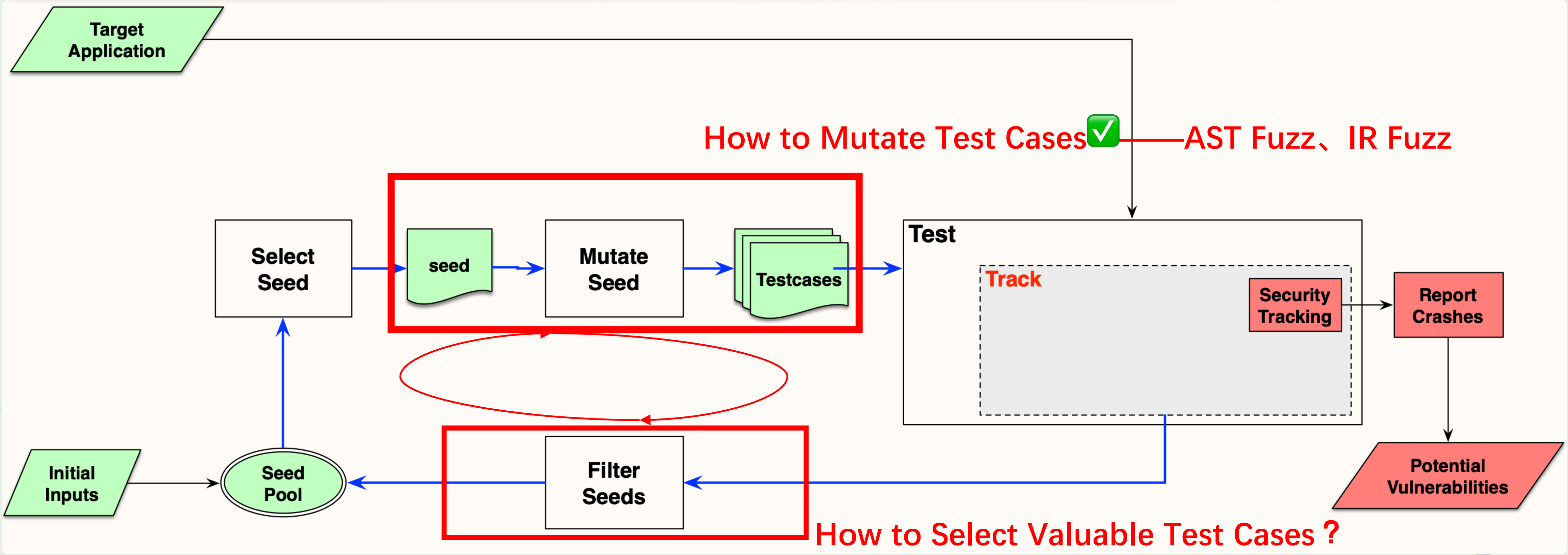
为特定类型设置了预设的property/method白名单，以array为例



```
  static let jsArray = Type.object(ofGroup: "Array", withProperties: ["__proto__", "length", "constructor"], withMethods: ["concat", "copyWithin",
"fill", "find", "findIndex", "pop", "push", "reverse", "shift", "unshift", "slice", "sort", "splice", "includes", "indexOf", "keys", "entries",
"forEach", "filter", "map", "every", "some", "reduce", "reduceRight", "toString", "toLocaleString", "join", "lastIndexOf", "values", "flat",
"flatMap"])
```

AFL：Mutate testcases

1. 选取一批testcases作为初始seeds
2. 变异生成更多的testcases
3. 保留一些testcases等待下一轮变异

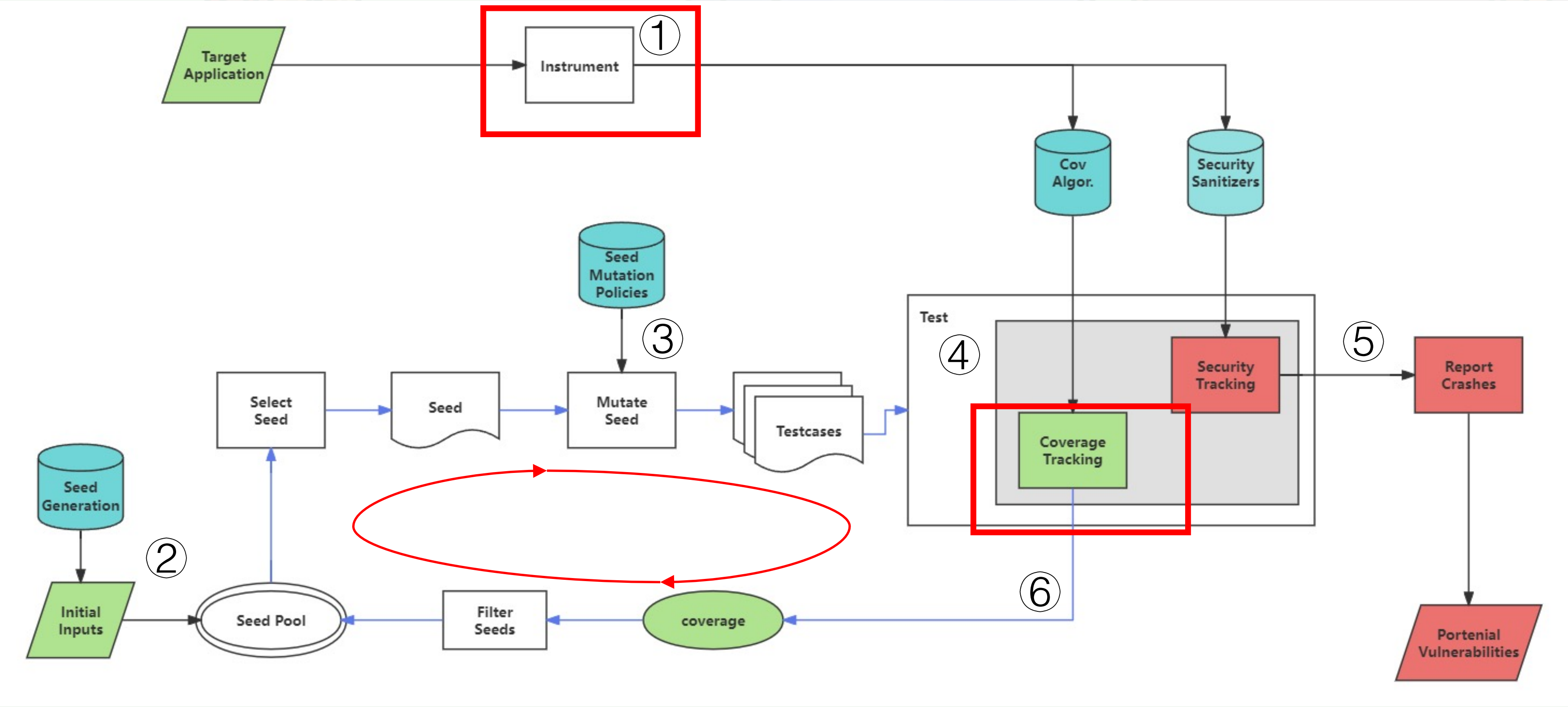


How to Mutate Test Cases✅——AST Fuzz、IR Fuzz

How to Select Valuable Test Cases？

AFL：Coverage-guided Fuzzing

触发了新路径的testcases被保留



工作流程

① 源码插桩

② 获取初始测试样例

③ 变异初始测试样例

④ 执行目标引擎

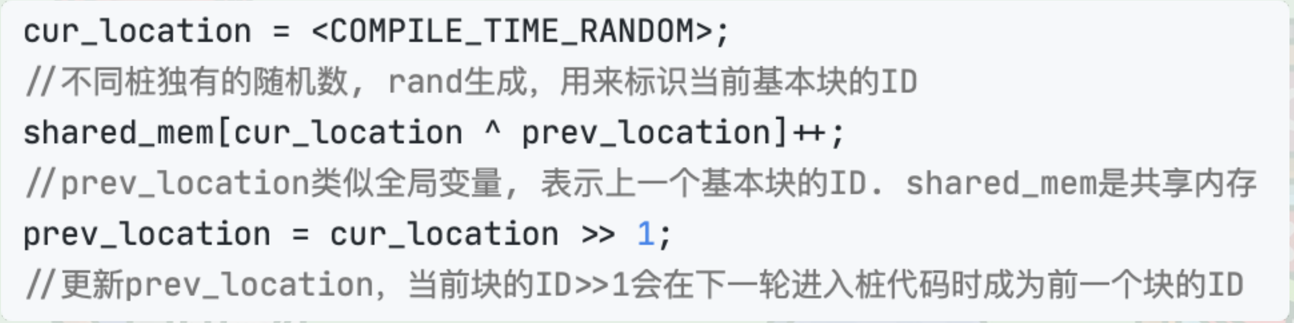
⑤ 记录crash

⑥ Coverage反馈

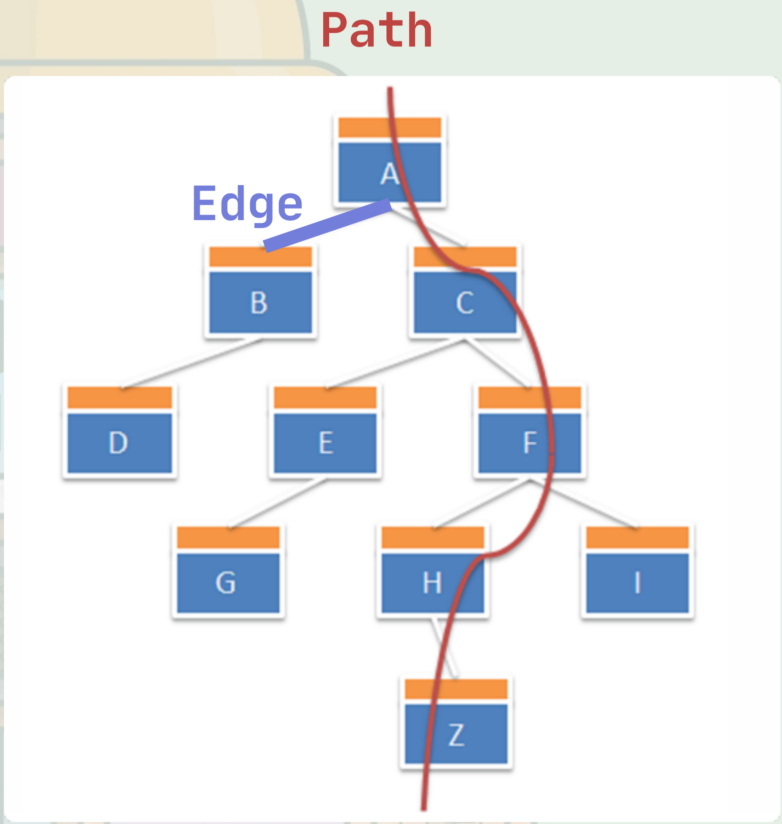
How to Instrument for Coverage Tracking？

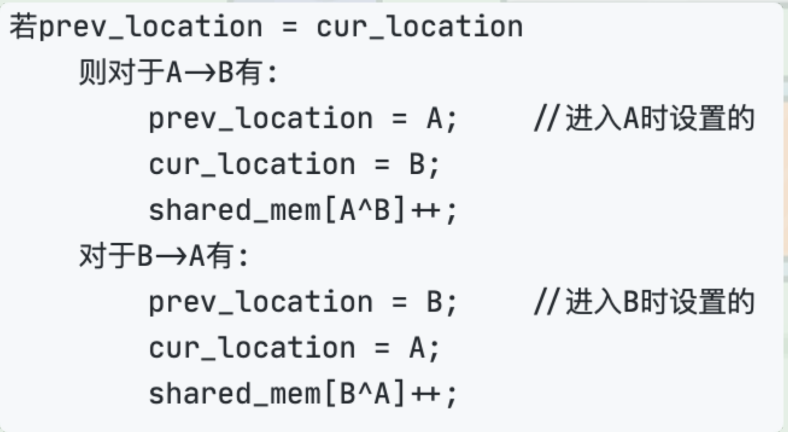
AFL ：Instrument——Edge Coverage

afl-gcc插入的桩代码核心部分

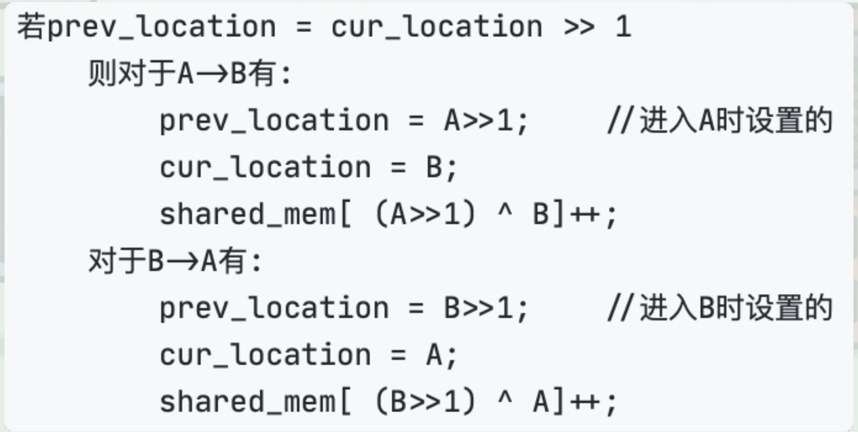


```
cur_location = <COMPILE_TIME_RANDOM>;
//不同桩独有的随机数，rand生成，用来标识当前基本块的ID
shared_mem[cur_location ^ prev_location]++;
//prev_location类似全局变量，表示上一个基本块的ID. shared_mem是共享内存
prev_location = cur_location >> 1;
//更新prev_location，当前块的ID>>1会在下一轮进入桩代码时成为前一个块的ID
```





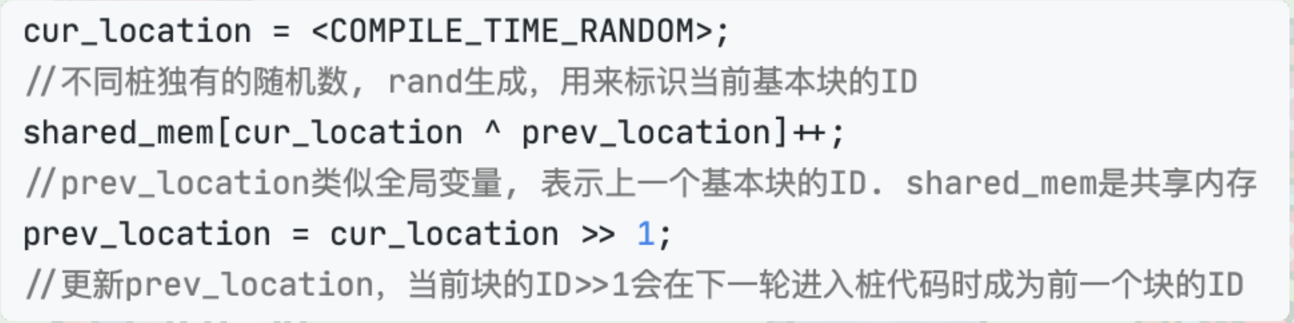
```
若prev_location = cur_location
    则对于A→B有:
        prev_location = A;    //进入A时设置的
        cur_location = B;
        shared_mem[A^B]++;
    对于B→A有:
        prev_location = B;    //进入B时设置的
        cur_location = A;
        shared_mem[B^A]++;
```



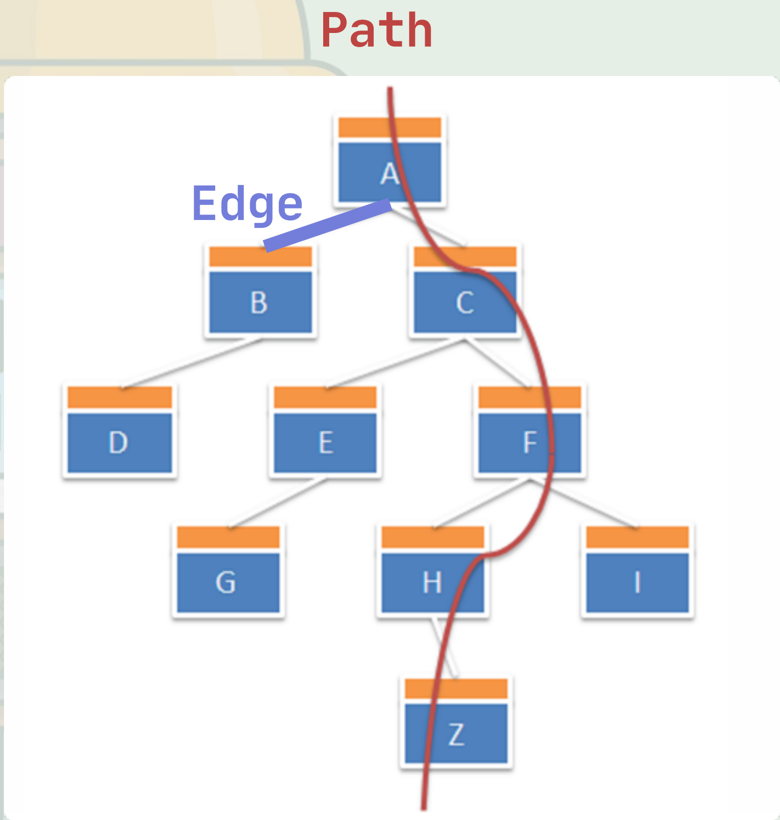
```
若prev_location = cur_location >> 1
    则对于A→B有:
        prev_location = A>>1;    //进入A时设置的
        cur_location = B;
        shared_mem[ (A>>1) ^ B]++;
    对于B→A有:
        prev_location = B>>1;    //进入B时设置的
        cur_location = A;
        shared_mem[ (B>>1) ^ A]++;
```

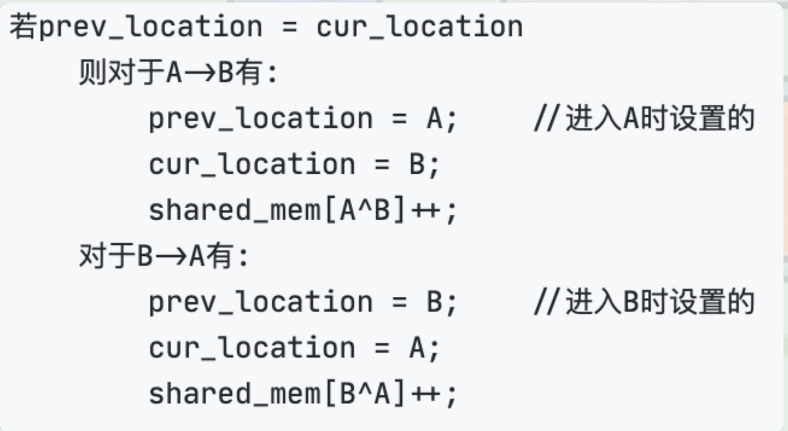
AFL：Instrument——Edge Coverage

afl-gcc插入的桩代码核心部分

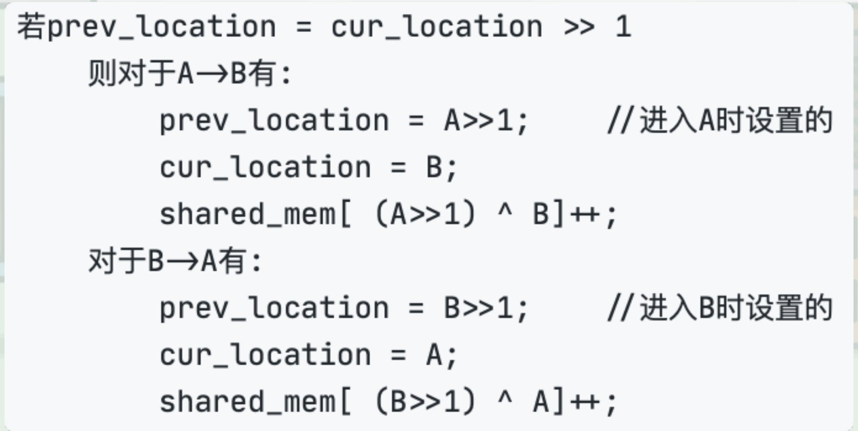


```
cur_location = <COMPILE_TIME_RANDOM>;
//不同桩独有的随机数，rand生成，用来标识当前基本块的ID
shared_mem[cur_location ^ prev_location]++;
//prev_location类似全局变量，表示上一个基本块的ID. shared_mem是共享内存
prev_location = cur_location >> 1;
//更新prev_location，当前块的ID>>1会在下一轮进入桩代码时成为前一个块的ID
```





```
若prev_location = cur_location
    则对于A→B有:
        prev_location = A;      //进入A时设置的
        cur_location = B;
        shared_mem[A^B]++;
    对于B→A有:
        prev_location = B;      //进入B时设置的
        cur_location = A;
        shared_mem[B^A]++;
```

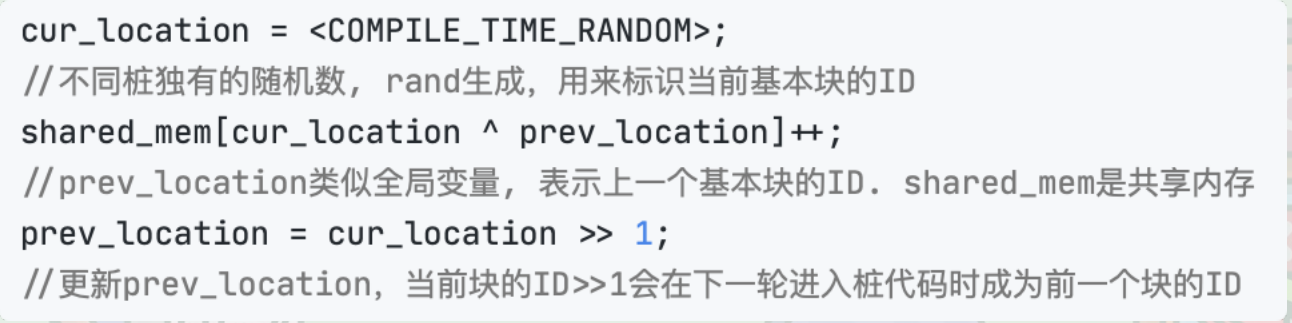


```
若prev_location = cur_location >> 1
    则对于A→B有:
        prev_location = A>>1;     //进入A时设置的
        cur_location = B;
        shared_mem[ (A>>1) ^ B]++;
    对于B→A有:
        prev_location = B>>1;     //进入B时设置的
        cur_location = A;
        shared_mem[ (B>>1) ^ A]++;
```

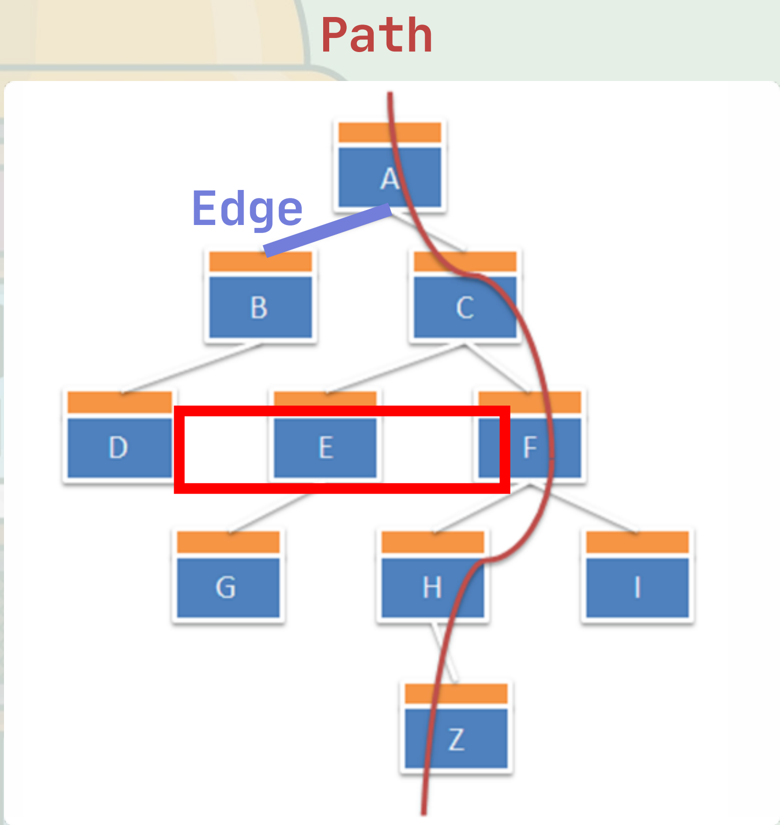
Insert Assembly(only x86) → Insert LLVM IR

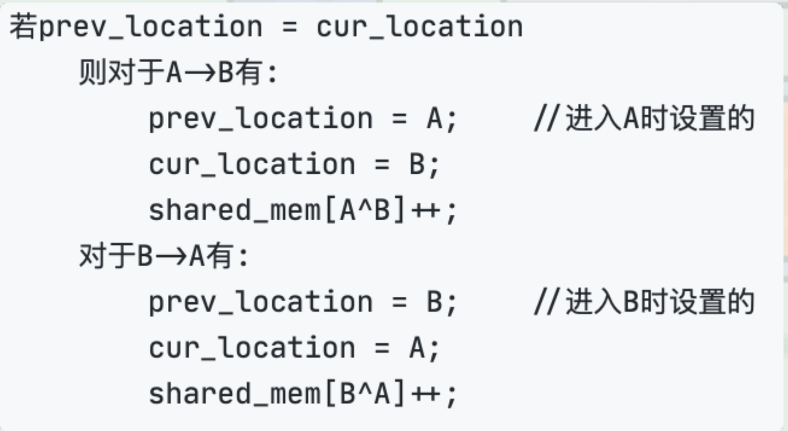
AFL：Instrument——Edge Coverage

afl-gcc插入的桩代码核心部分

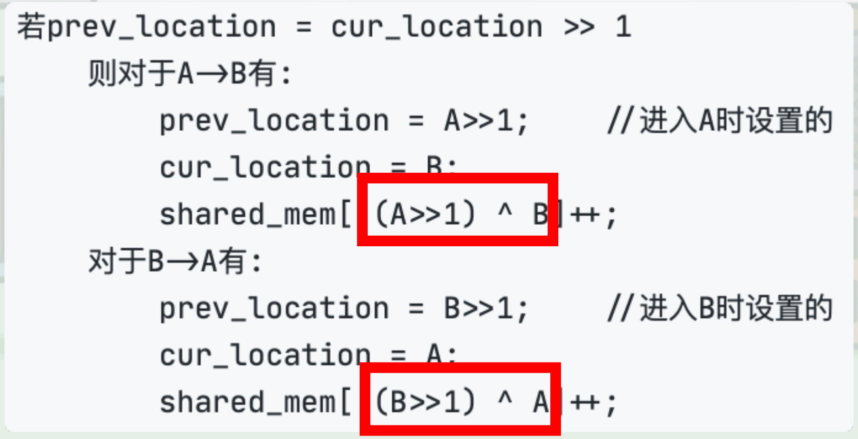


```
cur_location = <COMPILE_TIME_RANDOM>;
//不同桩独有的随机数，rand生成，用来标识当前基本块的ID
shared_mem[cur_location ^ prev_location]++;
//prev_location类似全局变量，表示上一个基本块的ID. shared_mem是共享内存
prev_location = cur_location >> 1;
//更新prev_location，当前块的ID>>1会在下一轮进入桩代码时成为前一个块的ID
```





```
若prev_location = cur_location
    则对于A—>B有:
        prev_location = A;    //进入A时设置的
        cur_location = B;
        shared_mem[A^B]++;
    对于B—>A有:
        prev_location = B;    //进入B时设置的
        cur_location = A;
        shared_mem[B^A]++;
```



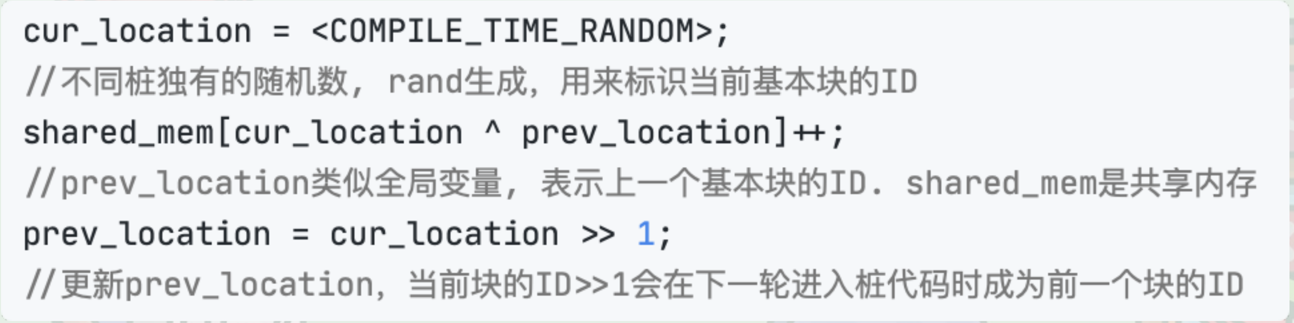
```
若prev_location = cur_location >> 1
    则对于A—>B有:
        prev_location = A>>1;    //进入A时设置的
        cur_location = B;
        shared_mem[(A>>1) ^ B]++;
    对于B—>A有:
        prev_location = B>>1;    //进入B时设置的
        cur_location = A;
        shared_mem[(B>>1) ^ A]++;
```

Hash Collision？

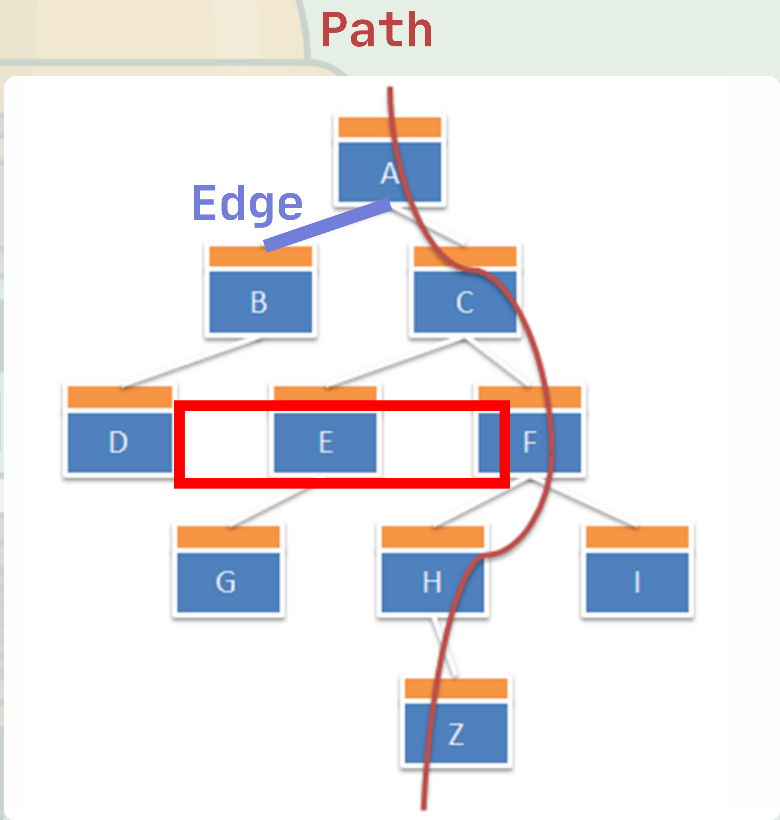
Insert Assembly(only x86) → Insert LLVM IR

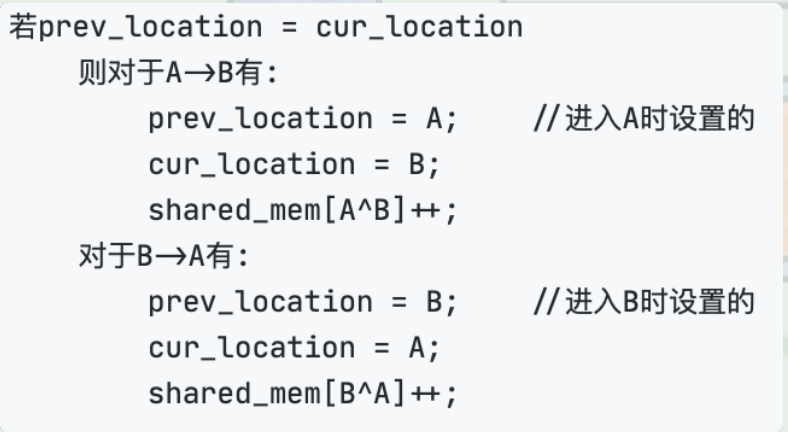
AFL：Instrument——Edge Coverage

afl-gcc插入的桩代码核心部分

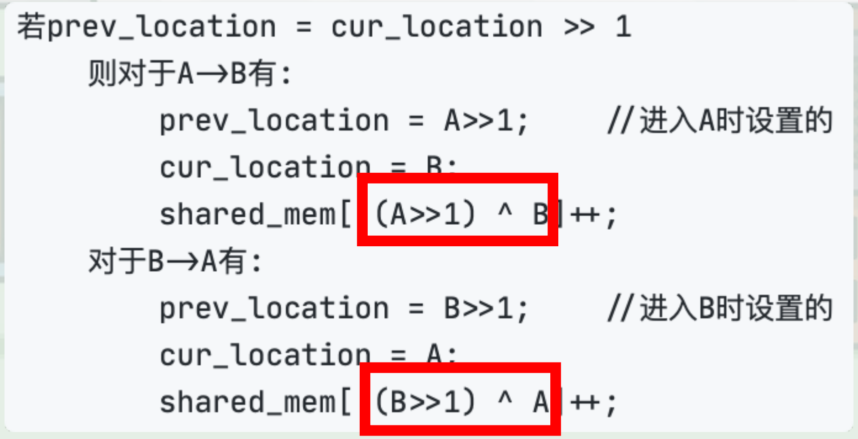


```
cur_location = <COMPILE_TIME_RANDOM>;
//不同桩独有的随机数，rand生成，用来标识当前基本块的ID
shared_mem[cur_location ^ prev_location]++;
//prev_location类似全局变量，表示上一个基本块的ID. shared_mem是共享内存
prev_location = cur_location >> 1;
//更新prev_location，当前块的ID>>1会在下一轮进入桩代码时成为前一个块的ID
```





```
若prev_location = cur_location
    则对于A—>B有:
        prev_location = A;     //进入A时设置的
        cur_location = B;
        shared_mem[A^B]++;
    对于B—>A有:
        prev_location = B;     //进入B时设置的
        cur_location = A;
        shared_mem[B^A]++;
```



```
若prev_location = cur_location >> 1
    则对于A—>B有:
        prev_location = A>>1;     //进入A时设置的
        cur_location = B;
        shared_mem[(A>>1) ^ B]++;
    对于B—>A有:
        prev_location = B>>1;     //进入B时设置的
        cur_location = A;
        shared_mem[(B>>1) ^ A]++;
```

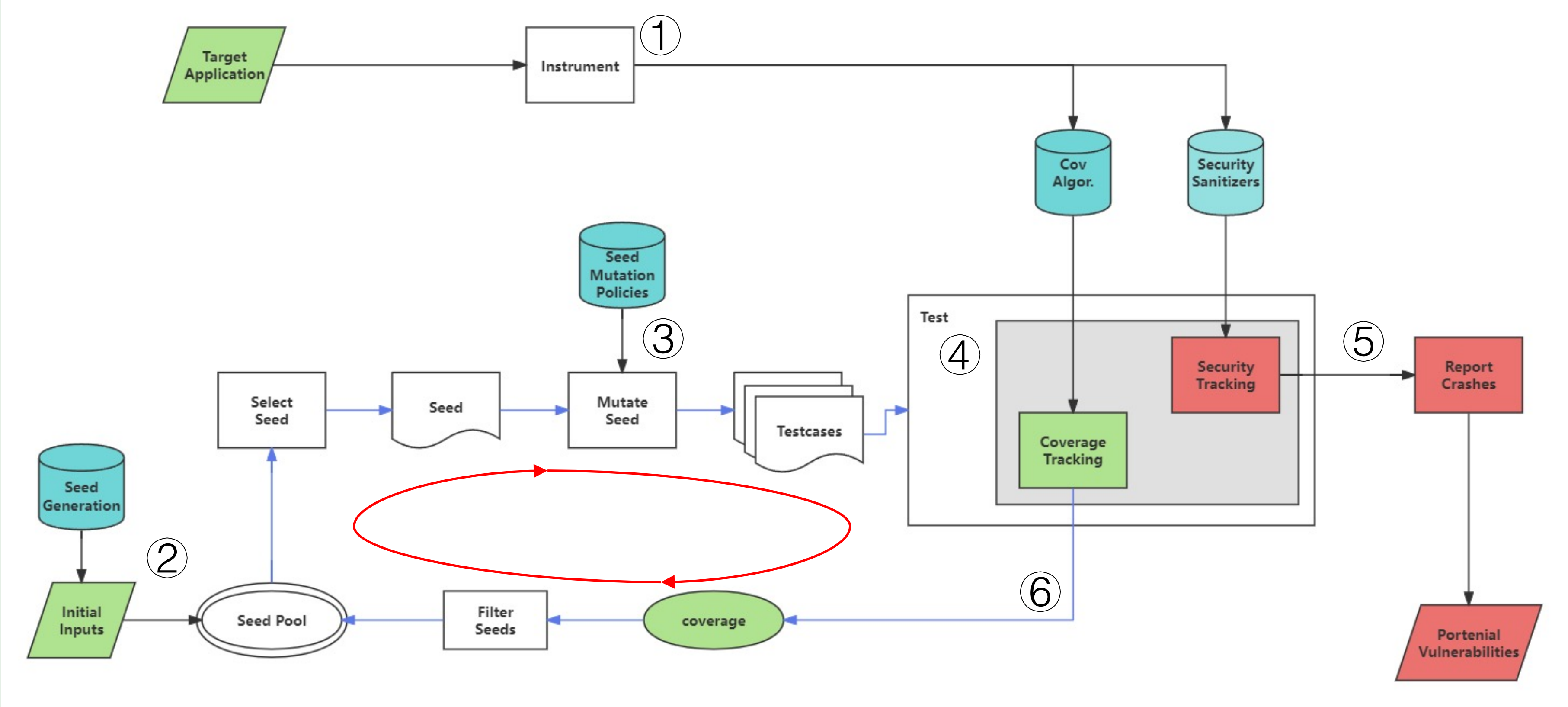
Hash Collision✅

Trace-pc-guard mode (LLVM Sanitizer Coverage) —— Each edge has a unique ID

AFL：Coverage-guided Fuzzing

How to Find Bugs More Effectively？

触发了新路径的testcases被保留



工作流程

① 源码插桩

② 获取初始测试样例

③ 变异初始测试样例

④ 执行目标引擎

⑤ 记录crash

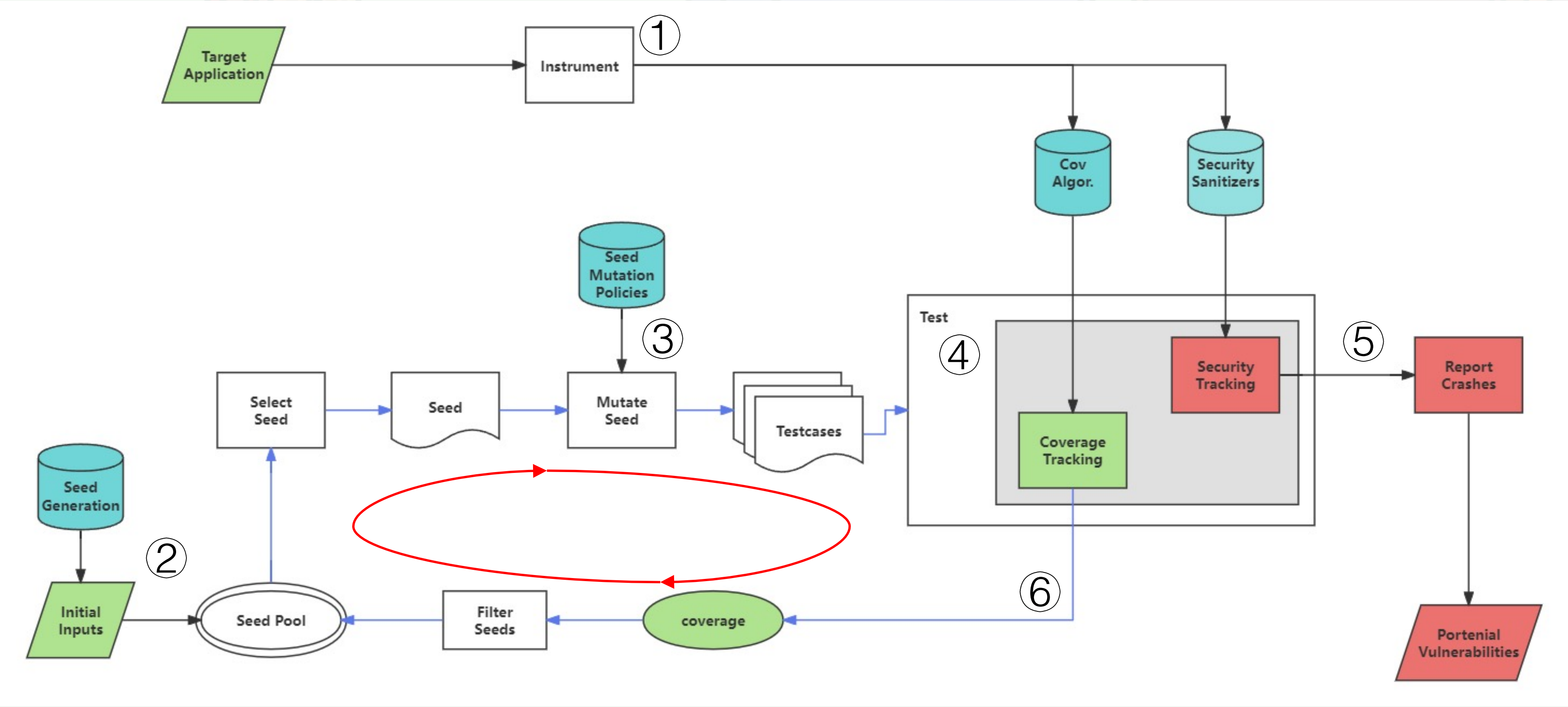
⑥ Coverage反馈

AFL：Coverage-guided Fuzzing

How to Find Bugs More Effectively✅

Fuzzing the JIT Component👾

触发了新路径的testcases被保留



工作流程

① 源码插桩

② 获取初始测试样例

③ 变异初始测试样例

④ 执行目标引擎

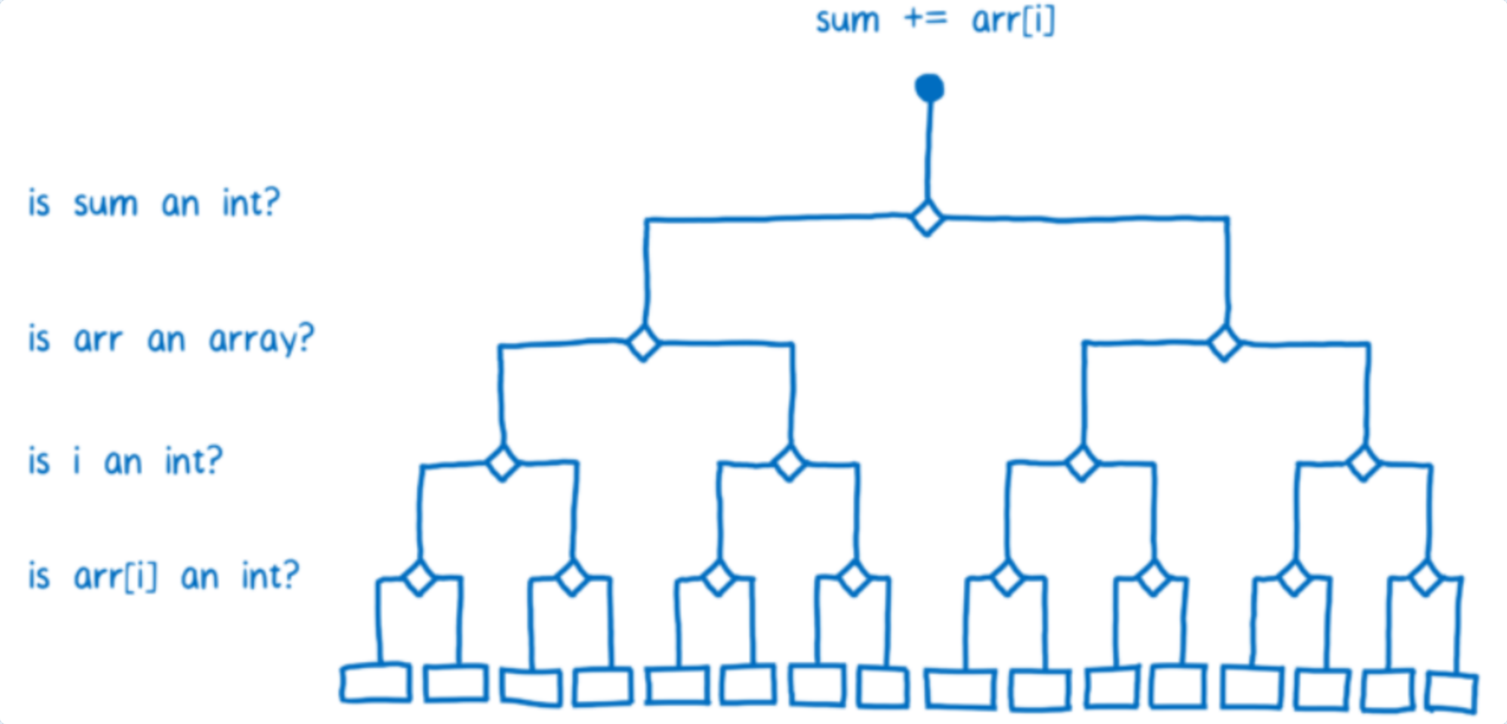
⑤ 记录crash

⑥ Coverage反馈

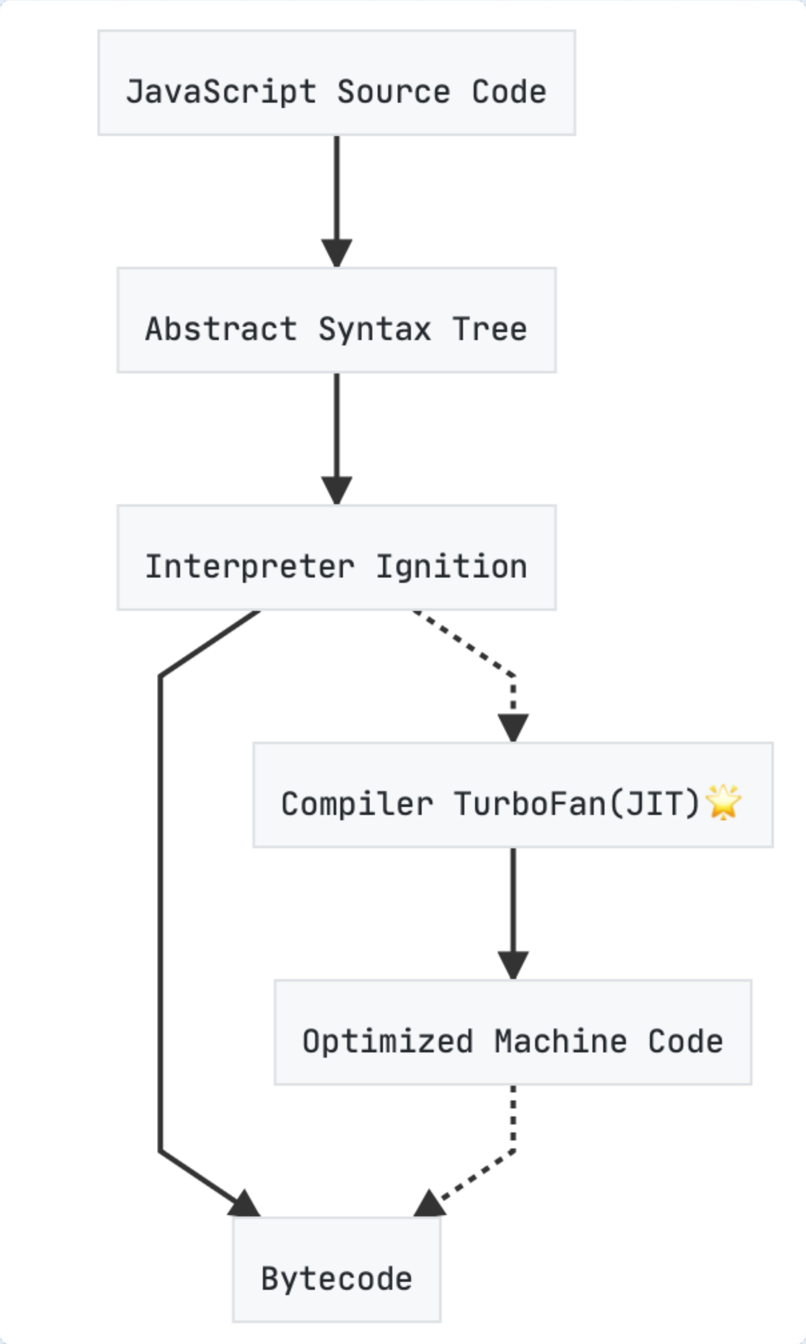
Background： JIT

JavaScript

- 动态类型语言，变量只有在运行时才能确定类型



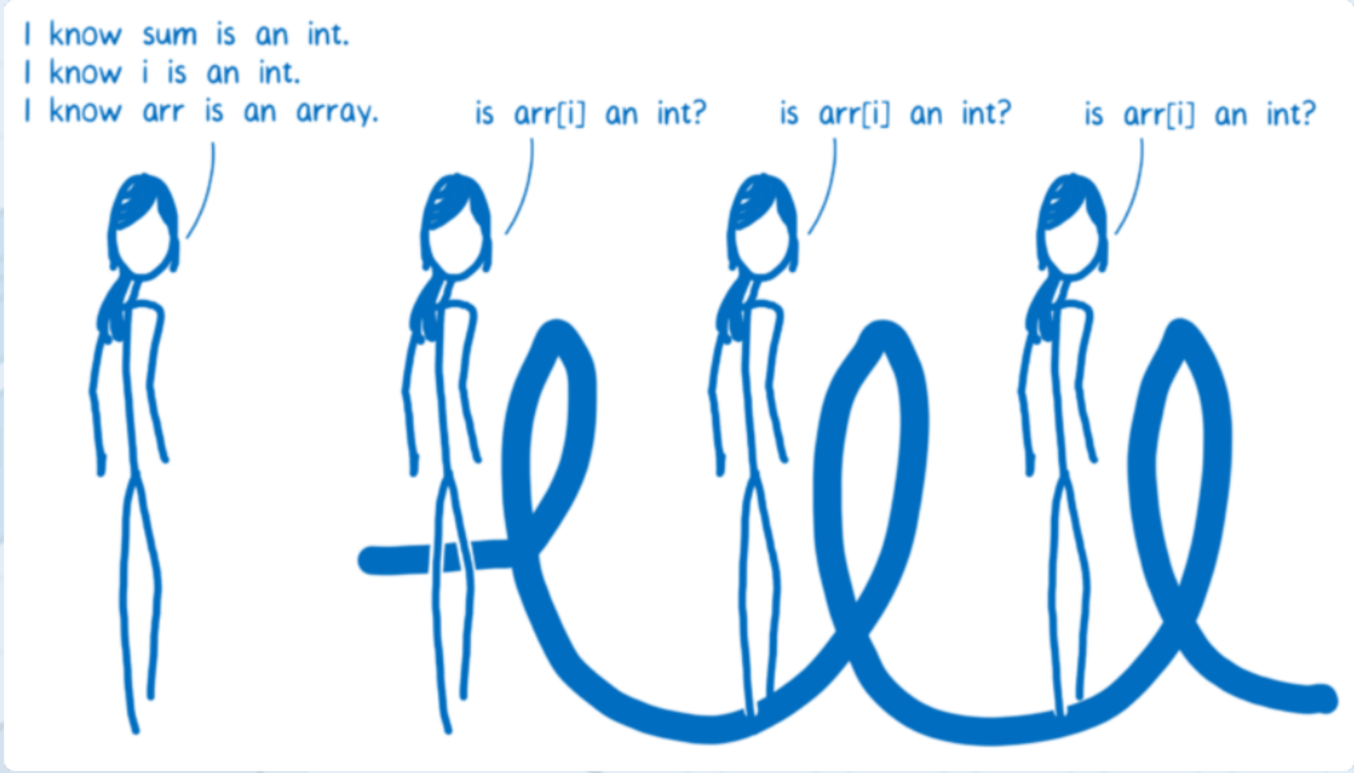
类型检查



Background： JIT

JIT

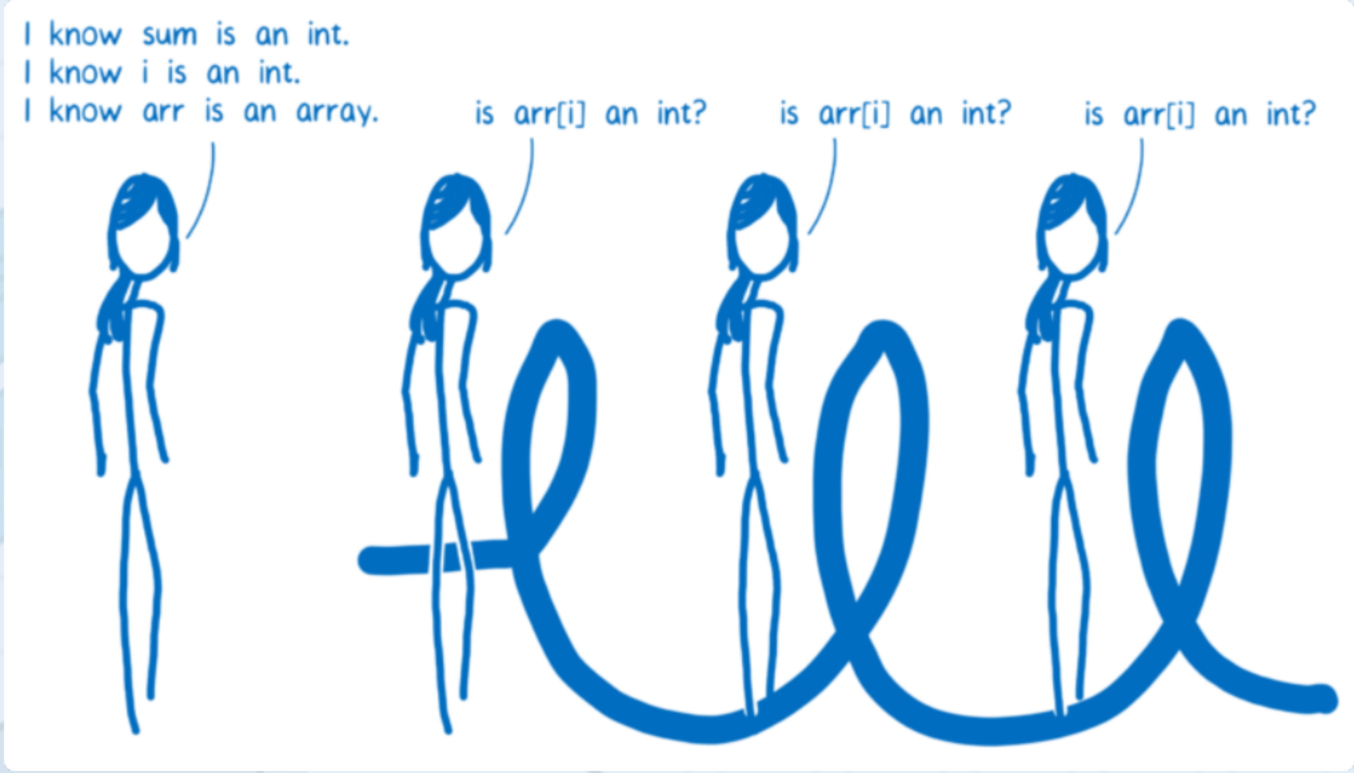
- 对这些变量的类型做出假设
- 删除冗余的类型检查
- 上保险🔒（类型保护）



Background： JIT

JIT如何做出合理假设？

- 根据 “概率”
- 存储多次执行的代码的编译结果
- 针对性的优化



Background： JIT

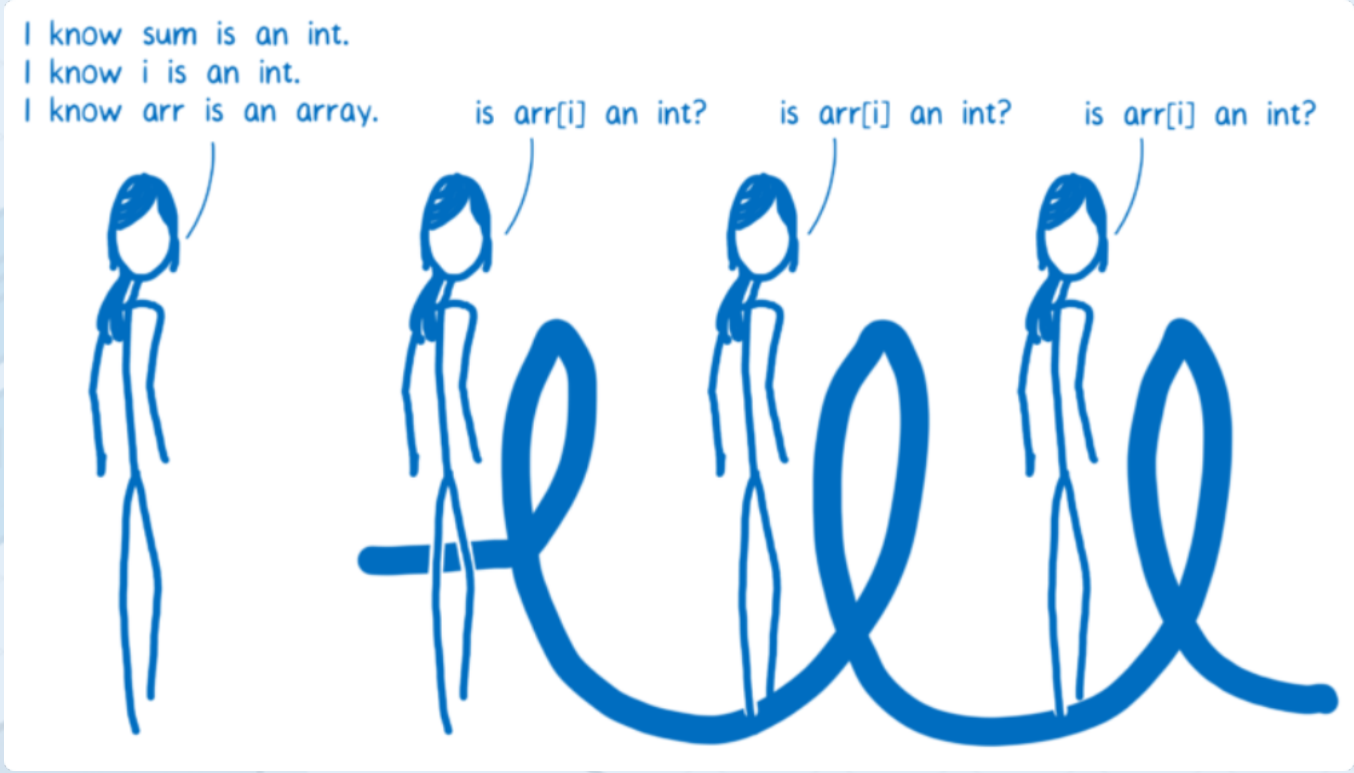
JIT

- 对这些变量的类型做出假设
- 删除冗余的类型检查
- 上保险🔒（类型保护）

Sometimes,~~🔒~~

↓

A vulnerability occurs



Background： JIT Vulnerability

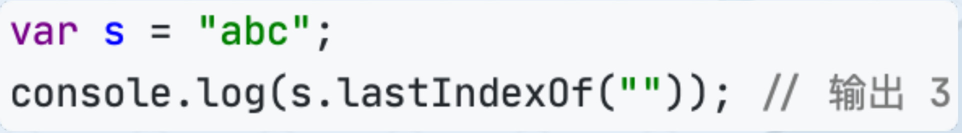
V8 off-by-one bug

V8数组越界访问漏洞

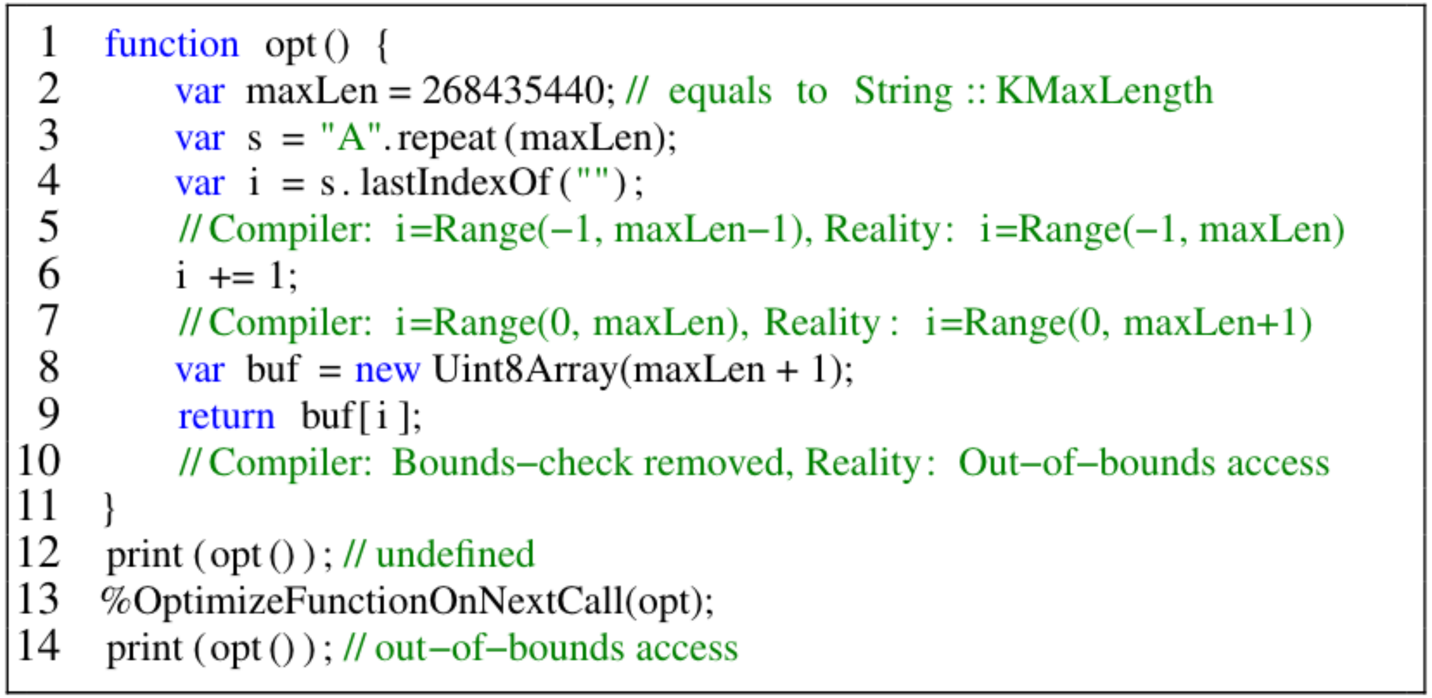
- Chrome的JS Engine
- JIT错误地消除了数组边界检查
- KMaxLength：String最大长度
- 越界读取一个元素⚠

原因

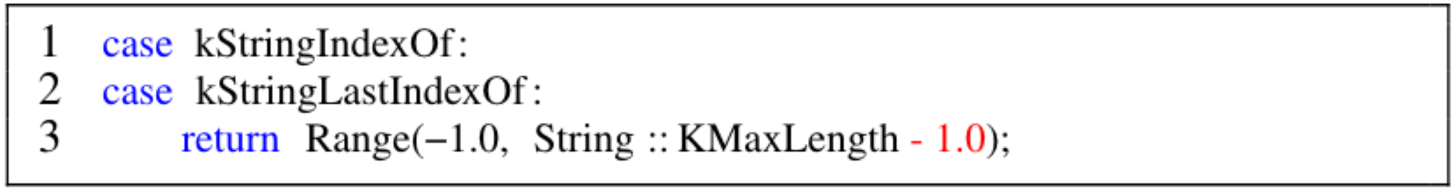
- JIT内部对string.lastIndexOf返回值范围设置为[-1,KMaxLength-1]
- lastIndexOf(“”)返回字符串长度



```
var s = "abc";
console.log(s.lastIndexOf("")); // 输出 3
```



```
function opt() {
    var maxLen = 268435440; // equals to String :: KMaxLength
    var s = "A".repeat(maxLen);
    var i = s.lastIndexOf("");
    //Compiler: i=Range(−1, maxLen−1), Reality: i=Range(−1, maxLen)
    i += 1;
    //Compiler: i=Range(0, maxLen), Reality : i=Range(0, maxLen+1)
    var buf = new Uint8Array(maxLen + 1);
    return buf[i];
    //Compiler: Bounds−check removed, Reality: Out−of−bounds access
}
print(opt()); // undefined
%OptimizeFunctionOnNextCall(opt);
print(opt()); // out−of−bounds access
```



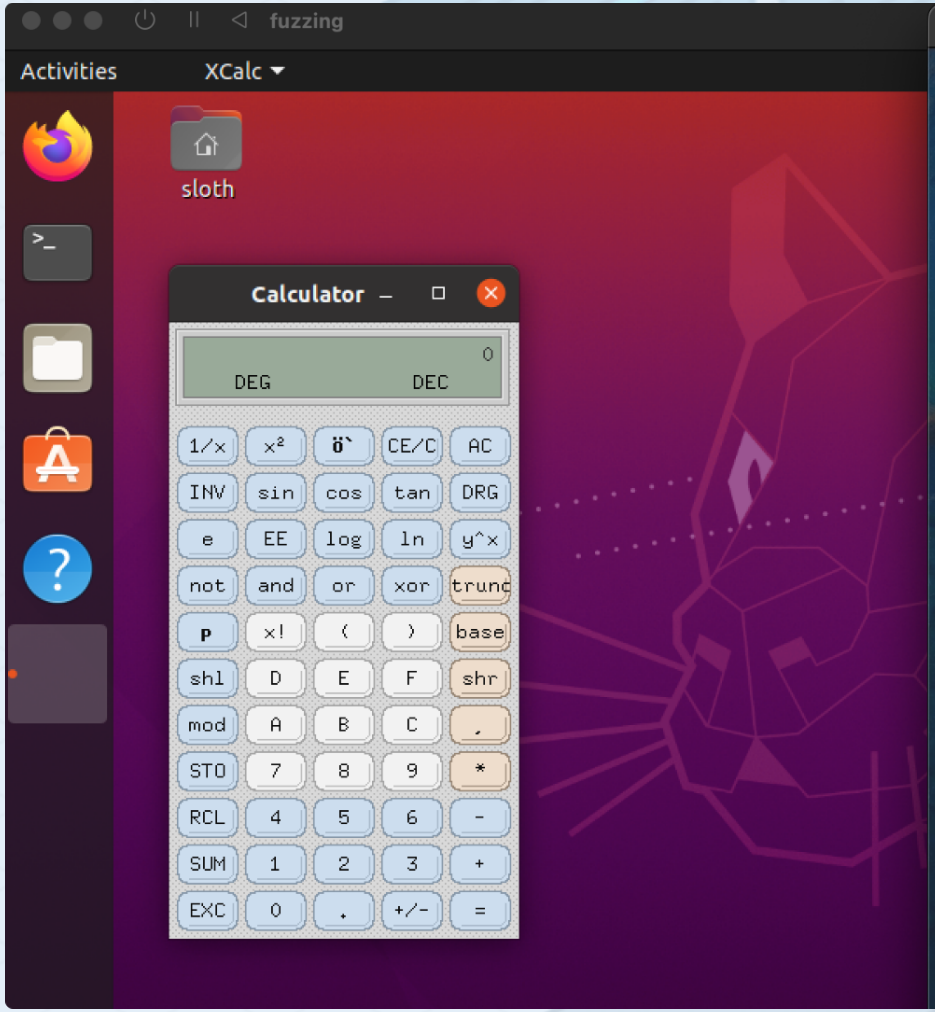
```
case kStringIndexOf:
case kStringLastIndexOf:
    return Range(−1.0, String :: KMaxLength - 1.0);
```

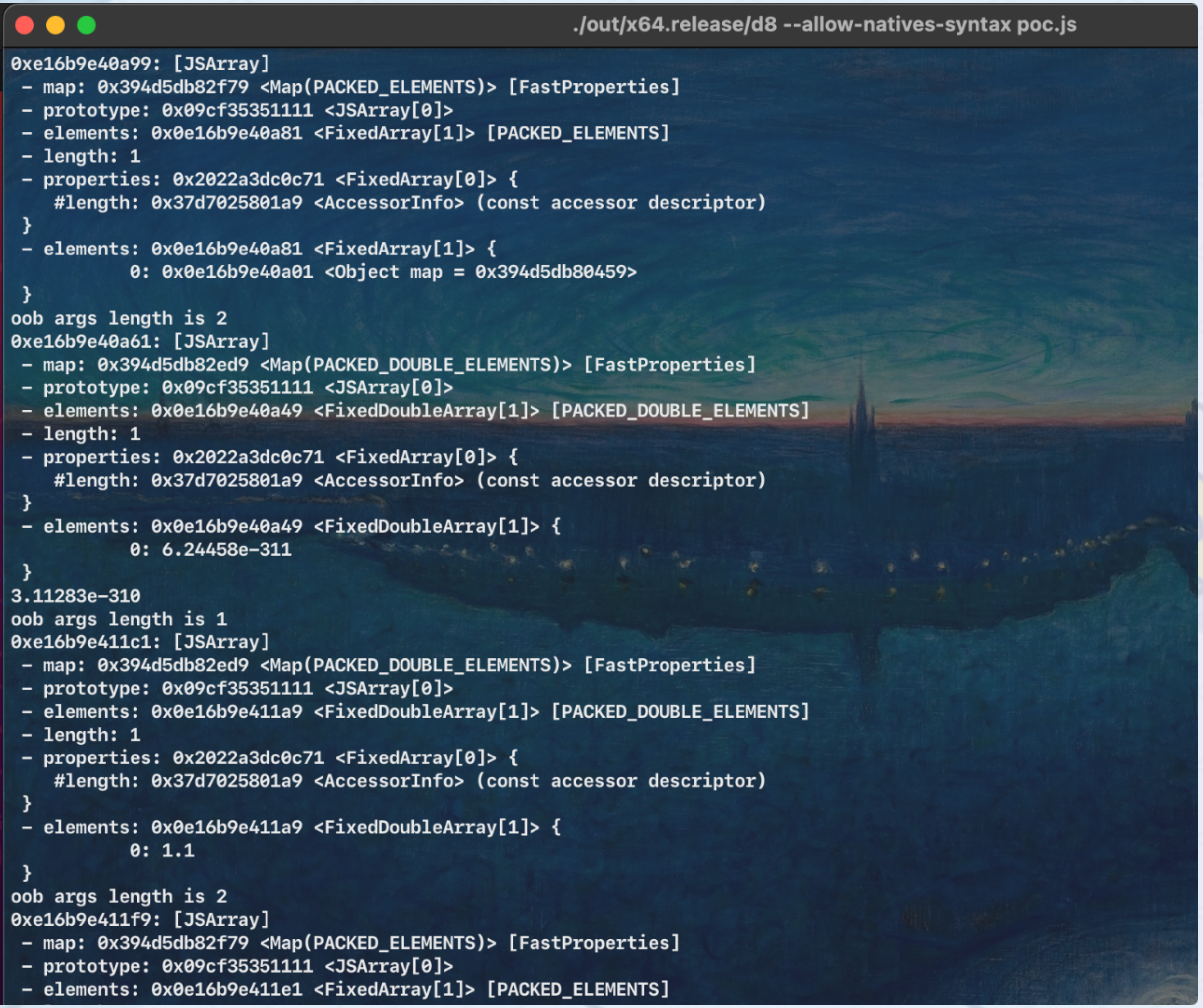
JIT实现相关源码

Background： JIT Vulnerability——V8 pwn

An OOB vulnerability can be exploited!

$ /usr/bin/calx



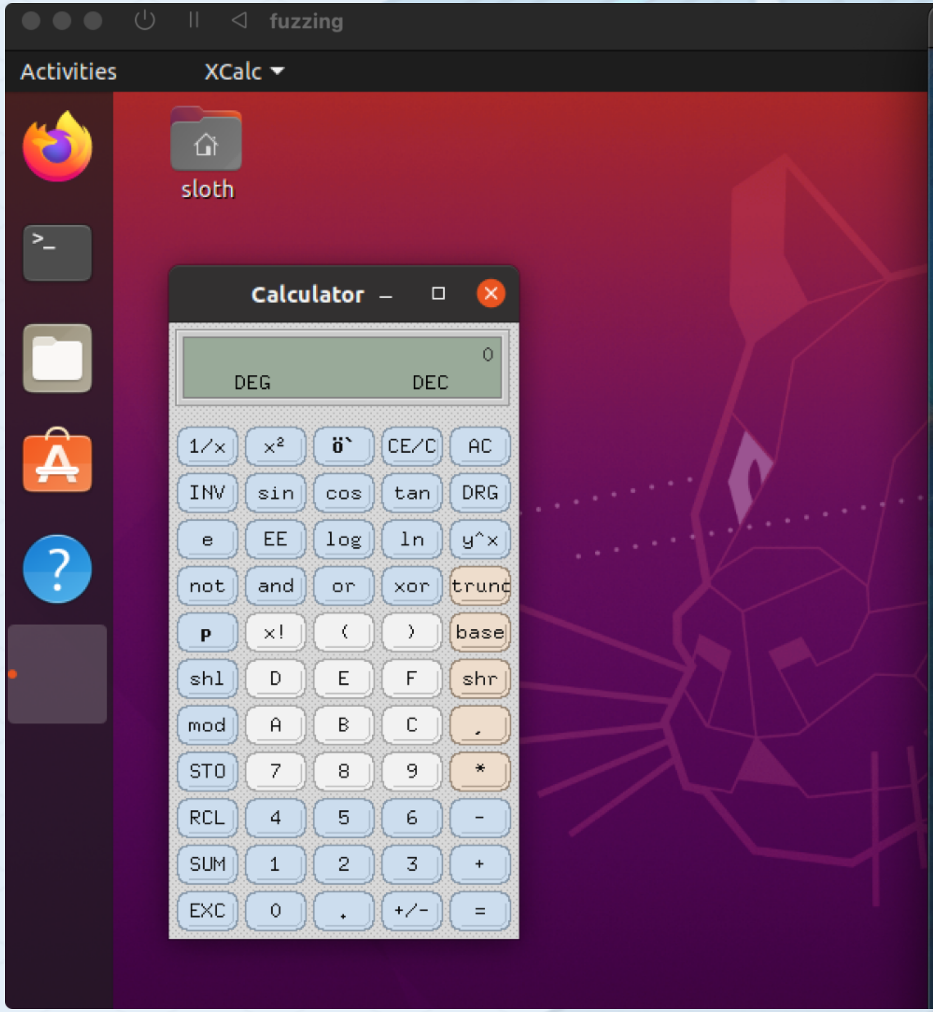

./out/x64.release/d8 --allow-natives-syntax poc.js

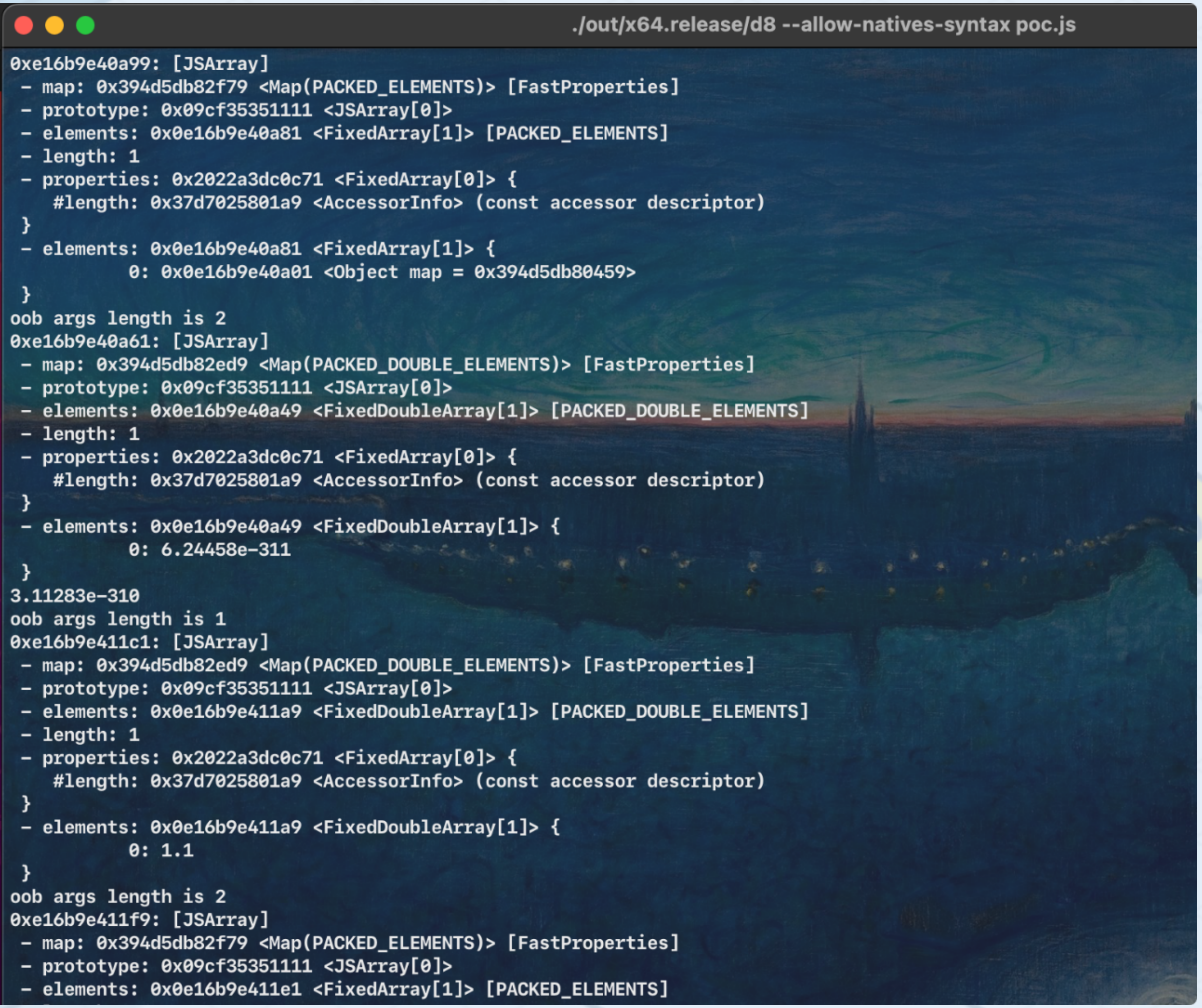
```
0xe16b9e40a99: [JSArray]
 - map: 0x394d5db82f79 <Map(PACKED_ELEMENTS)> [FastProperties]
 - prototype: 0x09cf35351111 <JSArray[0]>
 - elements: 0x0e16b9e40a81 <FixedArray[1]> [PACKED_ELEMENTS]
 - length: 1
 - properties: 0x2022a3dc0c71 <FixedArray[0]> {
    #length: 0x37d7025801a9 <AccessorInfo> (const accessor descriptor)
 }
 - elements: 0x0e16b9e40a81 <FixedArray[1]> {
           0: 0x0e16b9e40a01 <Object map = 0x394d5db80459>
 }
oob args length is 2
0xe16b9e40a61: [JSArray]
 - map: 0x394d5db82ed9 <Map(PACKED_DOUBLE_ELEMENTS)> [FastProperties]
 - prototype: 0x09cf35351111 <JSArray[0]>
 - elements: 0x0e16b9e40a49 <FixedDoubleArray[1]> [PACKED_DOUBLE_ELEMENTS]
 - length: 1
 - properties: 0x2022a3dc0c71 <FixedArray[0]> {
    #length: 0x37d7025801a9 <AccessorInfo> (const accessor descriptor)
 }
 - elements: 0x0e16b9e40a49 <FixedDoubleArray[1]> {
           0: 6.24458e-311
 }
3.11283e-310
oob args length is 1
0xe16b9e411c1: [JSArray]
 - map: 0x394d5db82ed9 <Map(PACKED_DOUBLE_ELEMENTS)> [FastProperties]
 - prototype: 0x09cf35351111 <JSArray[0]>
 - elements: 0x0e16b9e411a9 <FixedDoubleArray[1]> [PACKED_DOUBLE_ELEMENTS]
 - length: 1
 - properties: 0x2022a3dc0c71 <FixedArray[0]> {
    #length: 0x37d7025801a9 <AccessorInfo> (const accessor descriptor)
 }
 - elements: 0x0e16b9e411a9 <FixedDoubleArray[1]> {
           0: 1.1
 }
oob args length is 2
0xe16b9e411f9: [JSArray]
 - map: 0x394d5db82f79 <Map(PACKED_ELEMENTS)> [FastProperties]
 - prototype: 0x09cf35351111 <JSArray[0]>
 - elements: 0x0e16b9e411e1 <FixedArray[1]> [PACKED_ELEMENTS]
```

Background： JIT Vulnerability——V8 pwn

An OOB vulnerability can be exploited!

[*CTF]oob





./out/x64.release/d8 --allow-natives-syntax poc.js

```
0xe16b9e40a99: [JSArray]
 - map: 0x394d5db82f79 <Map(PACKED_ELEMENTS)> [FastProperties]
 - prototype: 0x09cf35351111 <JSArray[0]>
 - elements: 0x0e16b9e40a81 <FixedArray[1]> [PACKED_ELEMENTS]
 - length: 1
 - properties: 0x2022a3dc0c71 <FixedArray[0]> {
    #length: 0x37d7025801a9 <AccessorInfo> (const accessor descriptor)
 }
 - elements: 0x0e16b9e40a81 <FixedArray[1]> {
           0: 0x0e16b9e40a01 <Object map = 0x394d5db80459>
 }
oob args length is 2
0xe16b9e40a61: [JSArray]
 - map: 0x394d5db82ed9 <Map(PACKED_DOUBLE_ELEMENTS)> [FastProperties]
 - prototype: 0x09cf35351111 <JSArray[0]>
 - elements: 0x0e16b9e40a49 <FixedDoubleArray[1]> [PACKED_DOUBLE_ELEMENTS]
 - length: 1
 - properties: 0x2022a3dc0c71 <FixedArray[0]> {
    #length: 0x37d7025801a9 <AccessorInfo> (const accessor descriptor)
 }
 - elements: 0x0e16b9e40a49 <FixedDoubleArray[1]> {
           0: 6.24458e-311
 }
3.11283e-310
oob args length is 1
0xe16b9e411c1: [JSArray]
 - map: 0x394d5db82ed9 <Map(PACKED_DOUBLE_ELEMENTS)> [FastProperties]
 - prototype: 0x09cf35351111 <JSArray[0]>
 - elements: 0x0e16b9e411a9 <FixedDoubleArray[1]> [PACKED_DOUBLE_ELEMENTS]
 - length: 1
 - properties: 0x2022a3dc0c71 <FixedArray[0]> {
    #length: 0x37d7025801a9 <AccessorInfo> (const accessor descriptor)
 }
 - elements: 0x0e16b9e411a9 <FixedDoubleArray[1]> {
           0: 1.1
 }
oob args length is 2
0xe16b9e411f9: [JSArray]
 - map: 0x394d5db82f79 <Map(PACKED_ELEMENTS)> [FastProperties]
 - prototype: 0x09cf35351111 <JSArray[0]>
 - elements: 0x0e16b9e411e1 <FixedArray[1]> [PACKED_ELEMENTS]
```

Background： JIT Vulnerability

How to fuzz JIT？

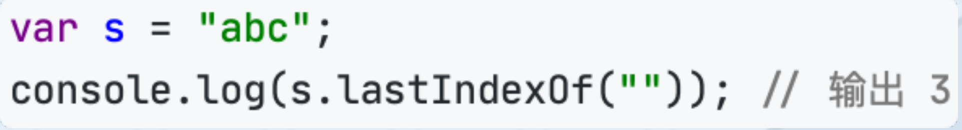
V8 off-by-one bug

V8数组越界访问漏洞

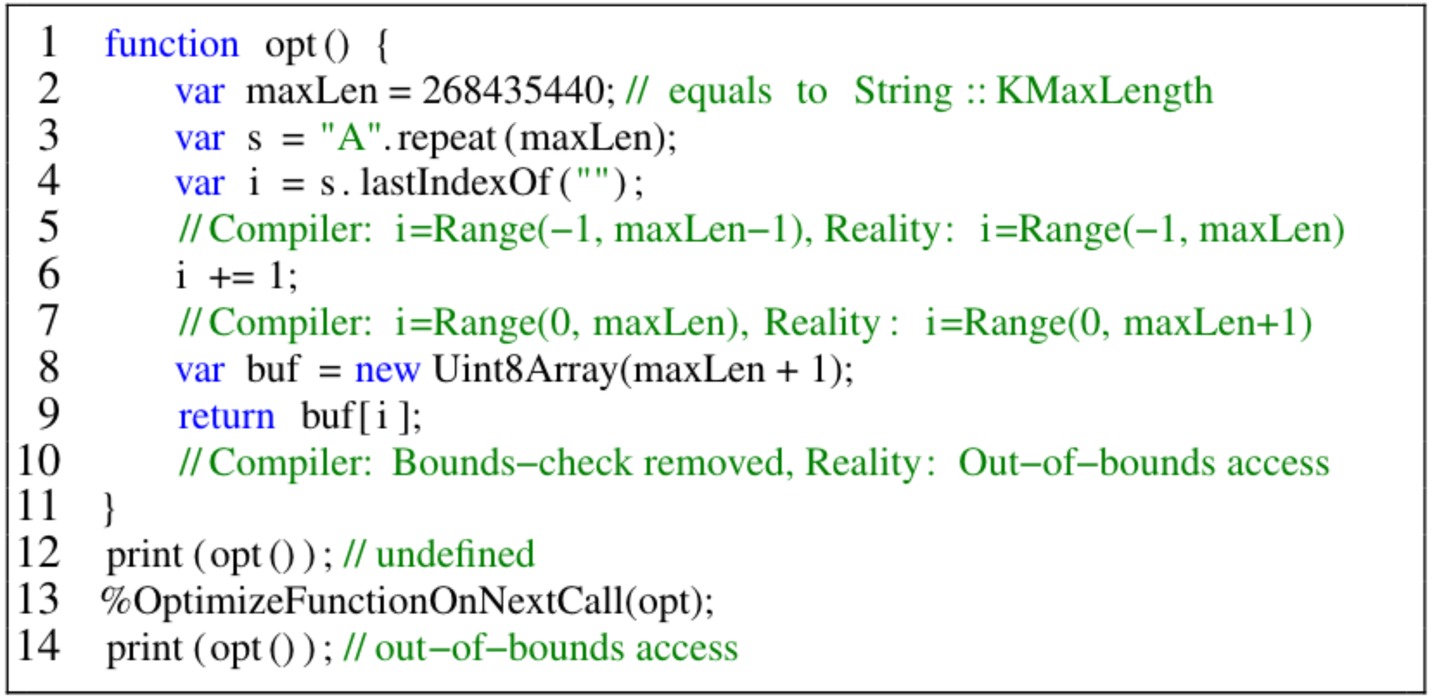
- Chrome的JS Engine
- JIT错误地消除了数组边界检查
- KMaxLength：String最大长度
- 越界读取一个元素⚠️

原因

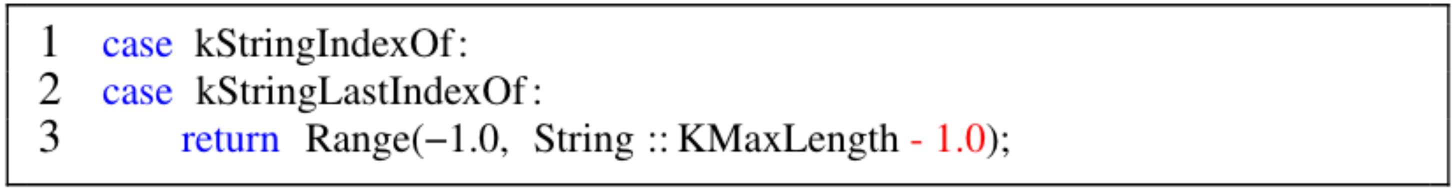
- JIT内部对string.lastIndexOf返回值范围设置为[-1,KMaxLength-1]
- lastIndexOf(“”)返回字符串长度



```
var s = "abc";
console.log(s.lastIndexOf("")); // 输出 3
```



```
function opt() {
    var maxLen = 268435440; // equals to String :: KMaxLength
    var s = "A".repeat(maxLen);
    var i = s.lastIndexOf("");
    //Compiler: i=Range(−1, maxLen−1), Reality: i=Range(−1, maxLen)
    i += 1;
    //Compiler: i=Range(0, maxLen), Reality : i=Range(0, maxLen+1)
    var buf = new Uint8Array(maxLen + 1);
    return buf[i];
    //Compiler: Bounds−check removed, Reality: Out−of−bounds access
}
print(opt()); // undefined
%OptimizeFunctionOnNextCall(opt);
print(opt()); // out−of−bounds access
```



```
case kStringIndexOf:
case kStringLastIndexOf:
    return Range(−1.0, String :: KMaxLength - 1.0);
```

JIT实现相关源码

FuzzJIT(USENIX Security'23): Background

chromium-issues bugs-chromium

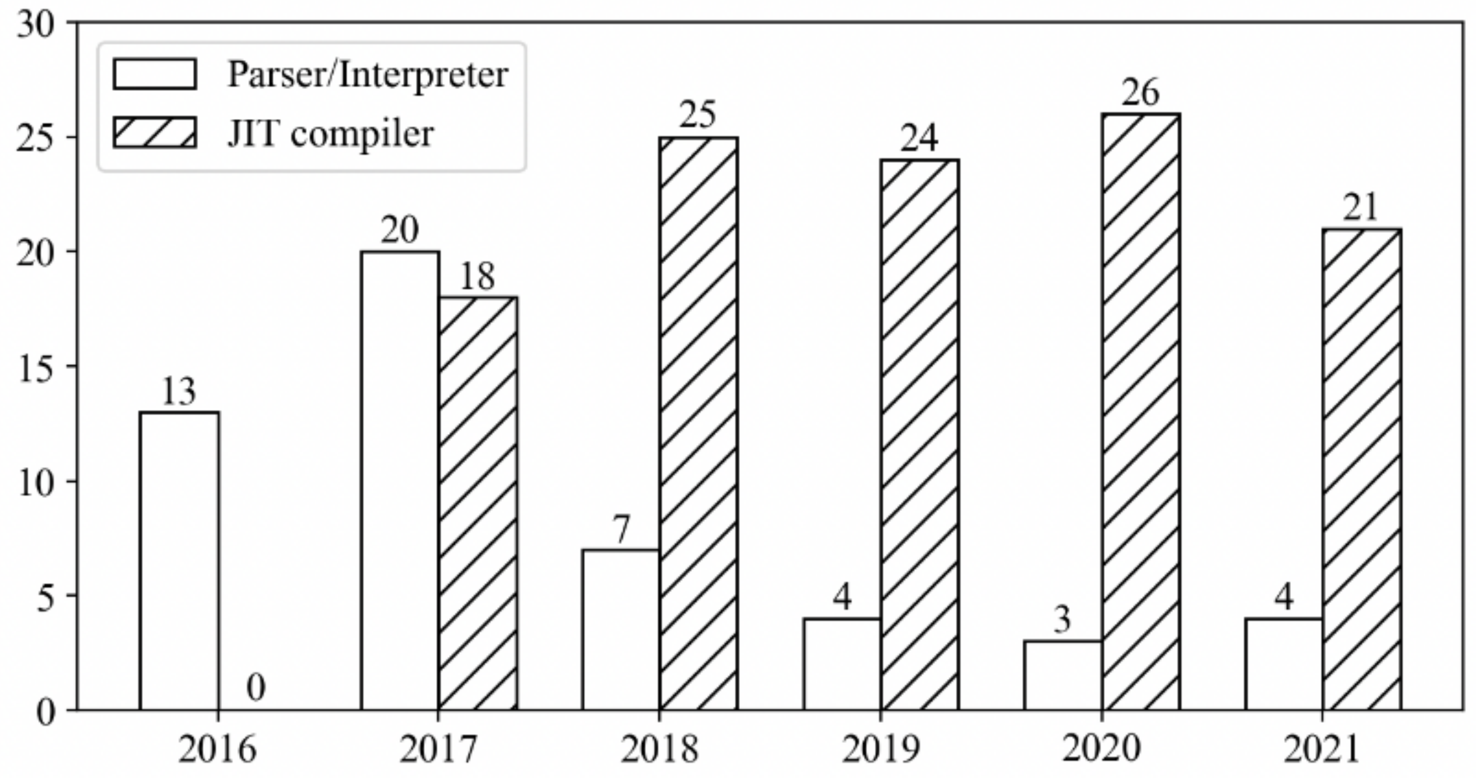


Figure 3: The number of bugs discovered respectively in parser/interpreter and in JIT compiler in recent years.

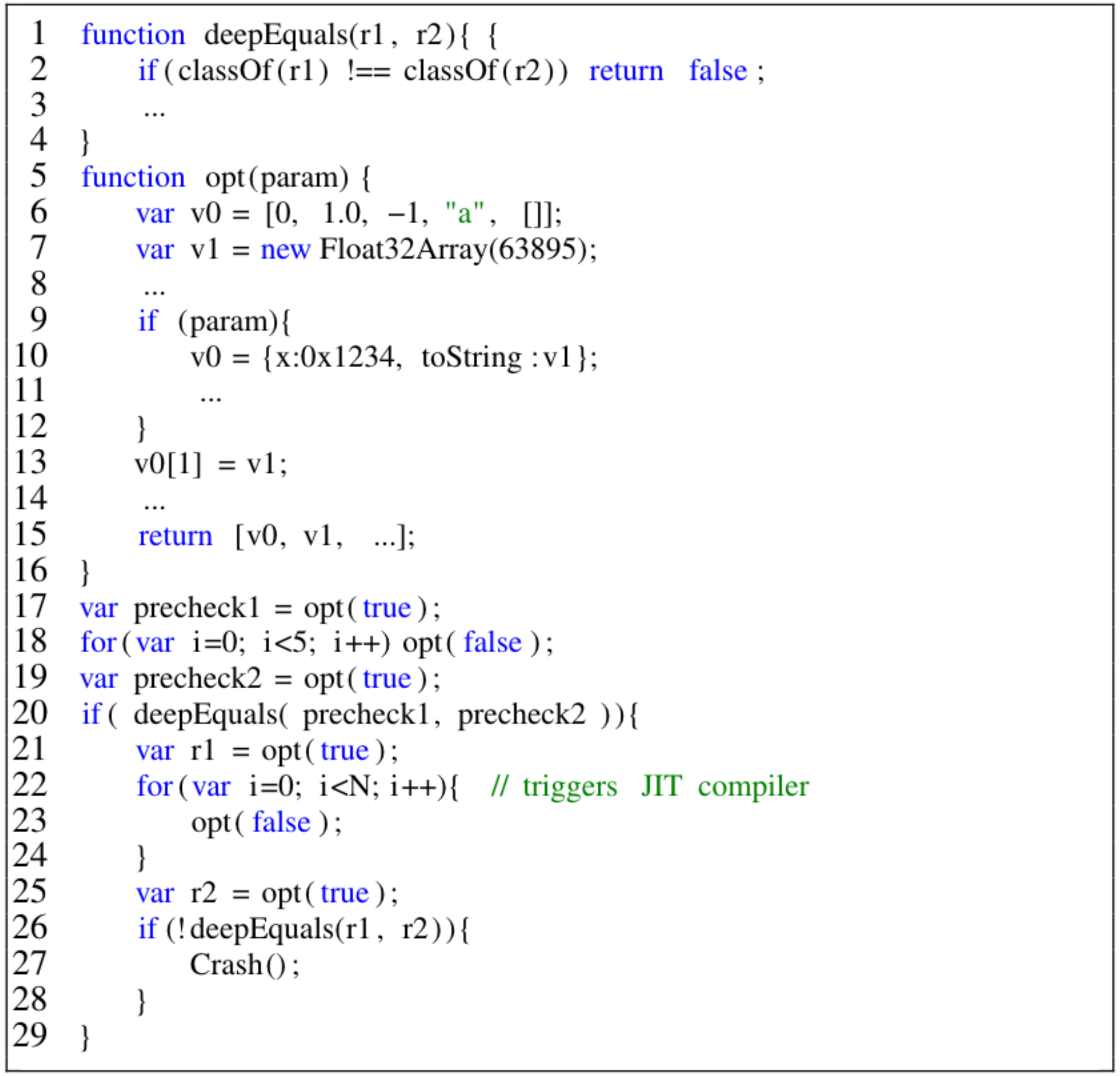
FuzzJIT(USENIX Security'23)

Fuzzilli

- CodeGenerators
  - 生成FuzzIL指令
- ProgramTemplates
  - 根据模版构建FuzzIL程序
- 支持自定义模版

FuzzJIT

- 基于Fuzzilli
- 提出一个fuzz JIT的模版

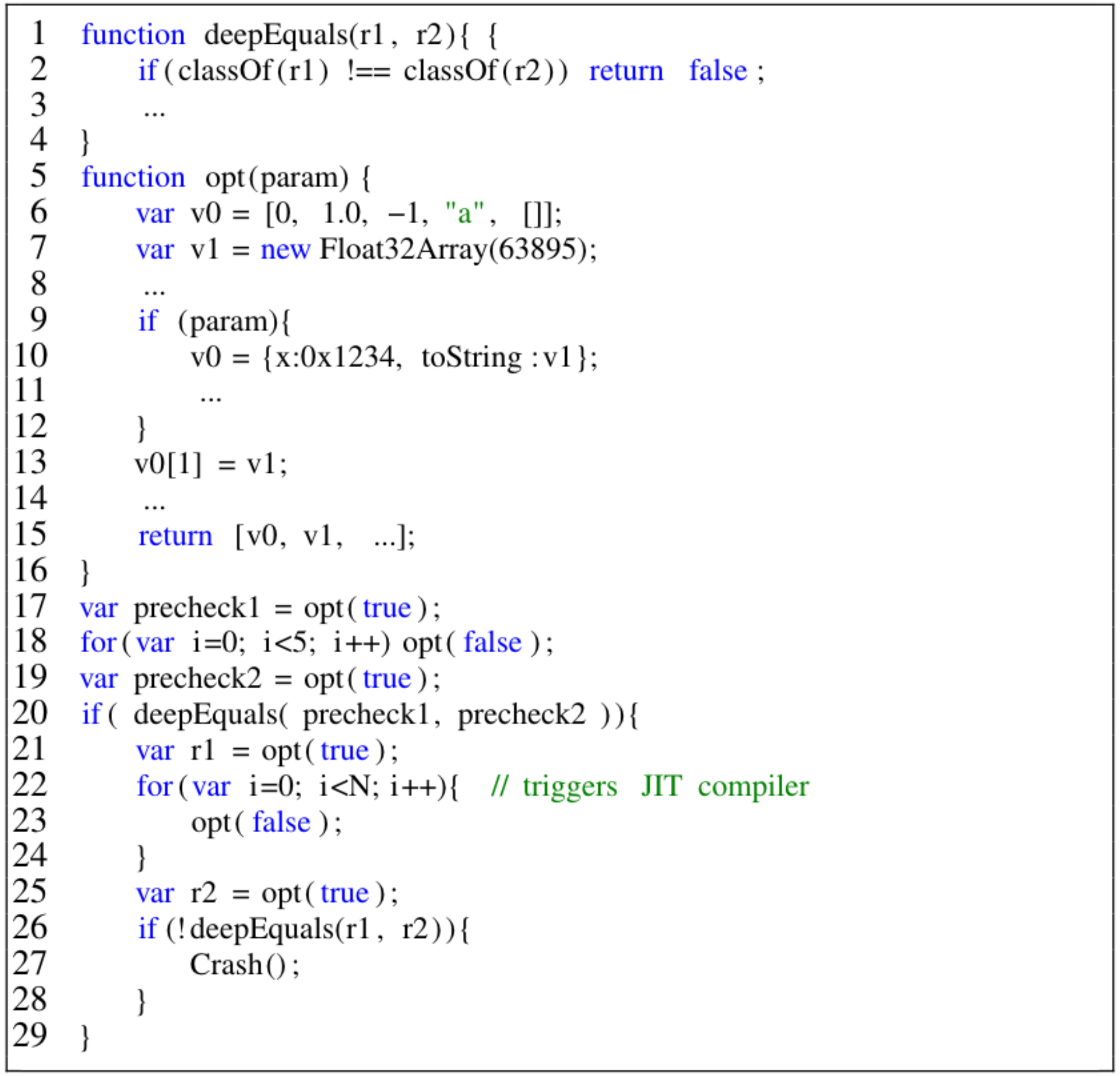


```
function deepEquals(r1, r2){ {
    if(classOf(r1) !== classOf(r2)) return false;
    ...
}
function opt(param) {
    var v0 = [0, 1.0, −1, "a", []];
    var v1 = new Float32Array(63895);
    ...
    if (param){
        v0 = {x:0x1234, toString:v1};
        ...
    }
    v0[1] = v1;
    ...
    return [v0, v1, ...];
}
var precheck1 = opt(true);
for(var i=0; i<5; i++) opt(false);
var precheck2 = opt(true);
if( deepEquals( precheck1, precheck2 )){
    var r1 = opt(true);
    for(var i=0; i<N; i++){ // triggers JIT compiler
        opt(false);
    }
    var r2 = opt(true);
    if (!deepEquals(r1, r2)){
        Crash();
    }
}
```

FuzzJIT(USENIX Security'23)

FuzzJIT模版

- opt
  - 封装主要测试的JS code
- Differential Testing
  - 利用deepEquals函数对比优化前后的输出
  - 如果不同，手动crash
- Triggers JIT
  - 构造循环



```
function deepEquals(r1, r2){ {
    if(classOf(r1) !== classOf(r2)) return false;
    ...
}
function opt(param) {
    var v0 = [0, 1.0, -1, "a", []];
    var v1 = new Float32Array(63895);
    ...
    if (param){
        v0 = {x:0x1234, toString:v1};
        ...
    }
    v0[1] = v1;
    ...
    return [v0, v1, ...];
}
var precheck1 = opt(true);
for(var i=0; i<5; i++) opt(false);
var precheck2 = opt(true);
if( deepEquals( precheck1, precheck2 )){
    var r1 = opt(true);
    for(var i=0; i<N; i++){  // triggers JIT compiler
        opt(false);
    }
    var r2 = opt(true);
    if(!deepEquals(r1, r2)){
        Crash();
    }
}
```

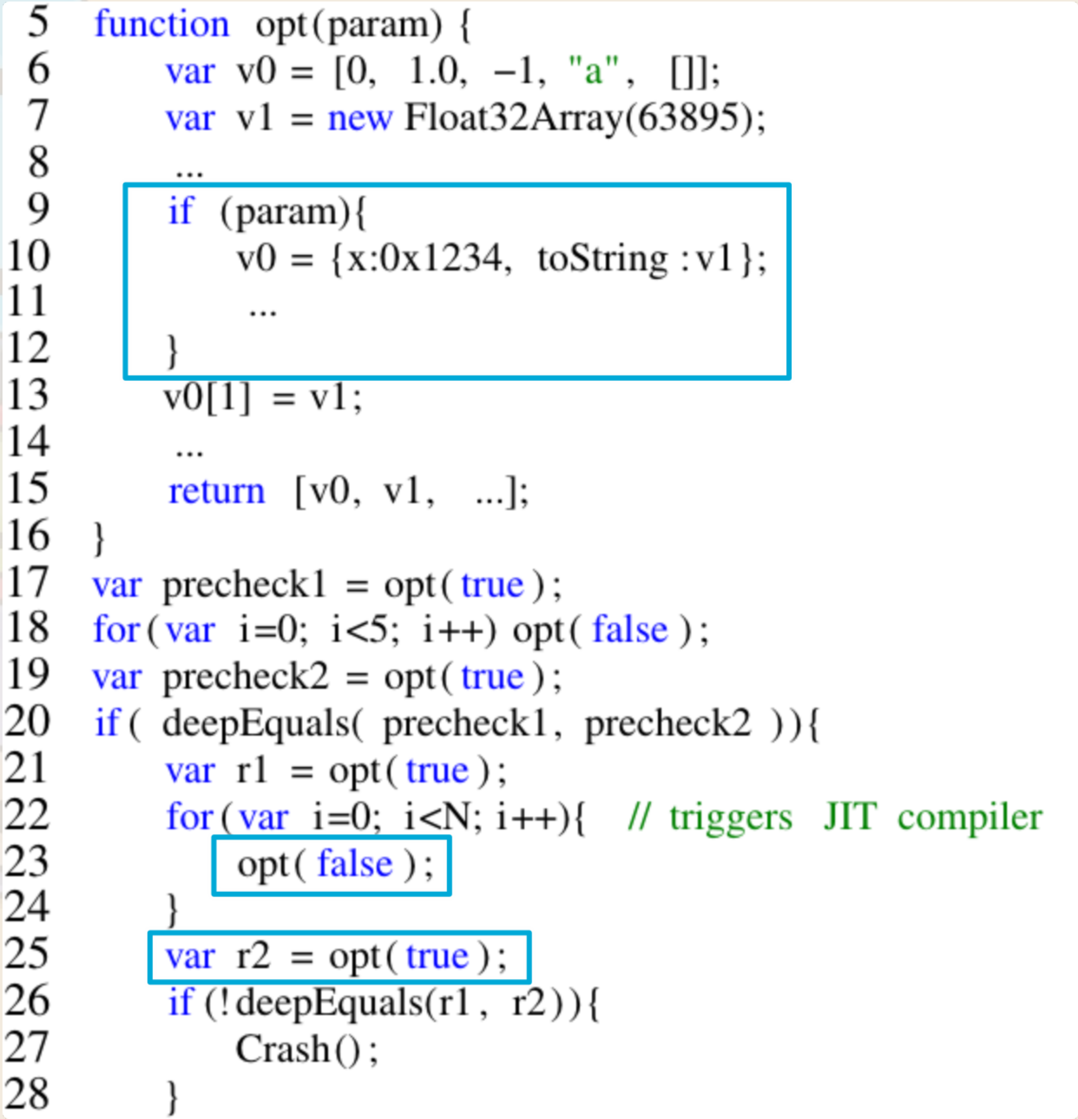
FuzzJIT：Revealing JIT compiler bugs

FuzzJIT主要测试的JIT优化方向

- 数组边界检查
- 变量类型检查
- 公共子表达式消除

Conditioned variable reassignments

- 触发JIT的循环中,param=false,"骗"JIT变量v0类型一直不变，使其把类型检查优化掉
- Triggers JIT后,param=true,"偷偷改变变量类型",增加触发类型混淆bug的可能性



```
function opt(param) {
    var v0 = [0, 1.0, −1, "a", []];
    var v1 = new Float32Array(63895);
    ...
    if (param){
        v0 = {x:0x1234, toString :v1};
        ...
    }
    v0[1] = v1;
    ...
    return [v0, v1, ...];
}
var precheck1 = opt(true);
for(var i=0; i<5; i++) opt(false);
var precheck2 = opt(true);
if( deepEquals( precheck1, precheck2 )){
    var r1 = opt(true);
    for(var i=0; i<N; i++){  // triggers JIT compiler
        opt(false);
    }
    var r2 = opt(true);
    if (!deepEquals(r1, r2)){
        Crash();
    }
```

FuzzJIT：Revealing JIT compiler bugs

deepEquals

- 比较优化前后的输出
- avoid false positives
  - API黑名单
  - 排除随机性
    - Math.random()
    - Data.now()
    - …

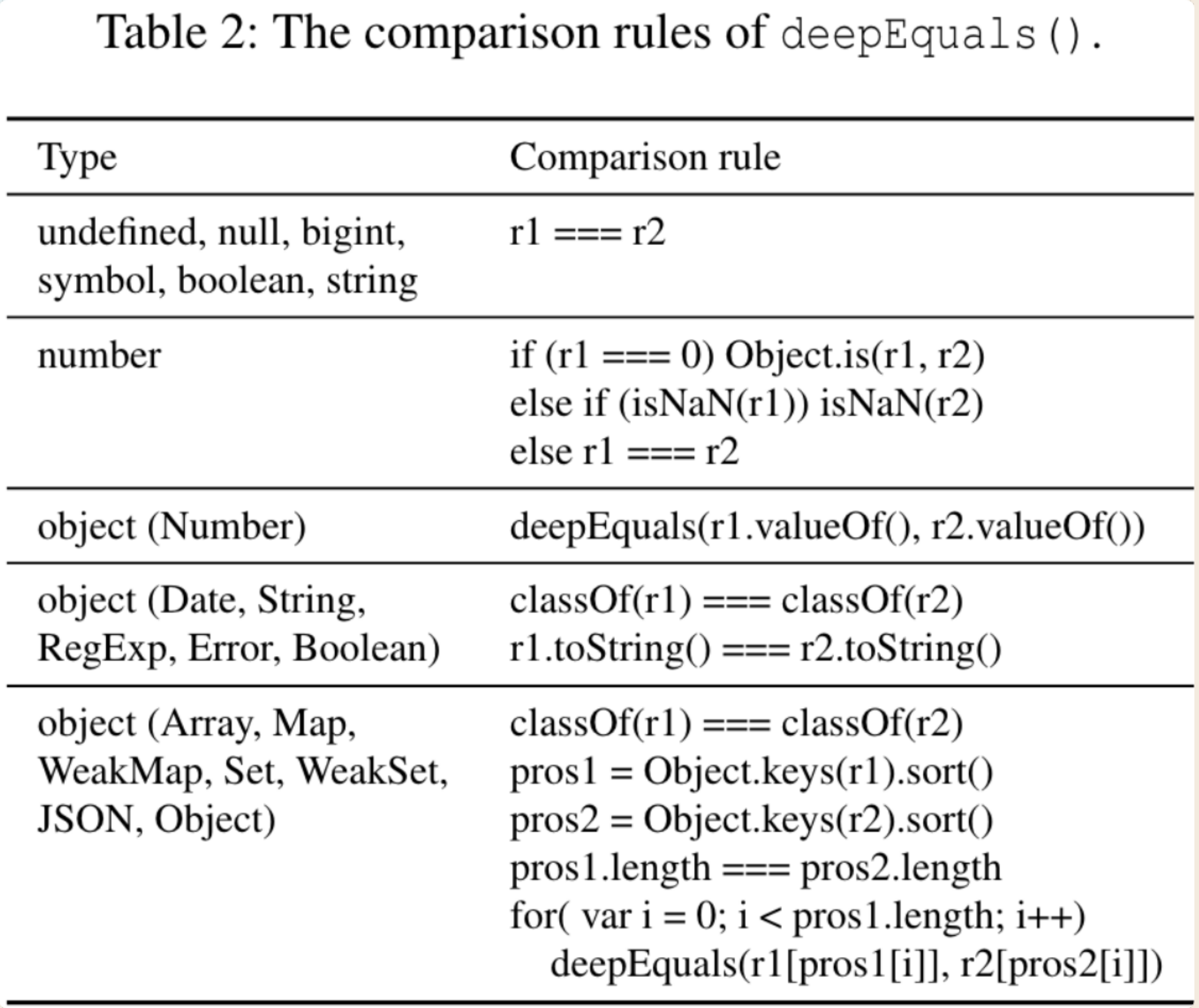


Table 2: The comparison rules of deepEquals().

Type	Comparison rule
undefined, null, bigint, symbol, boolean, string	r1 === r2

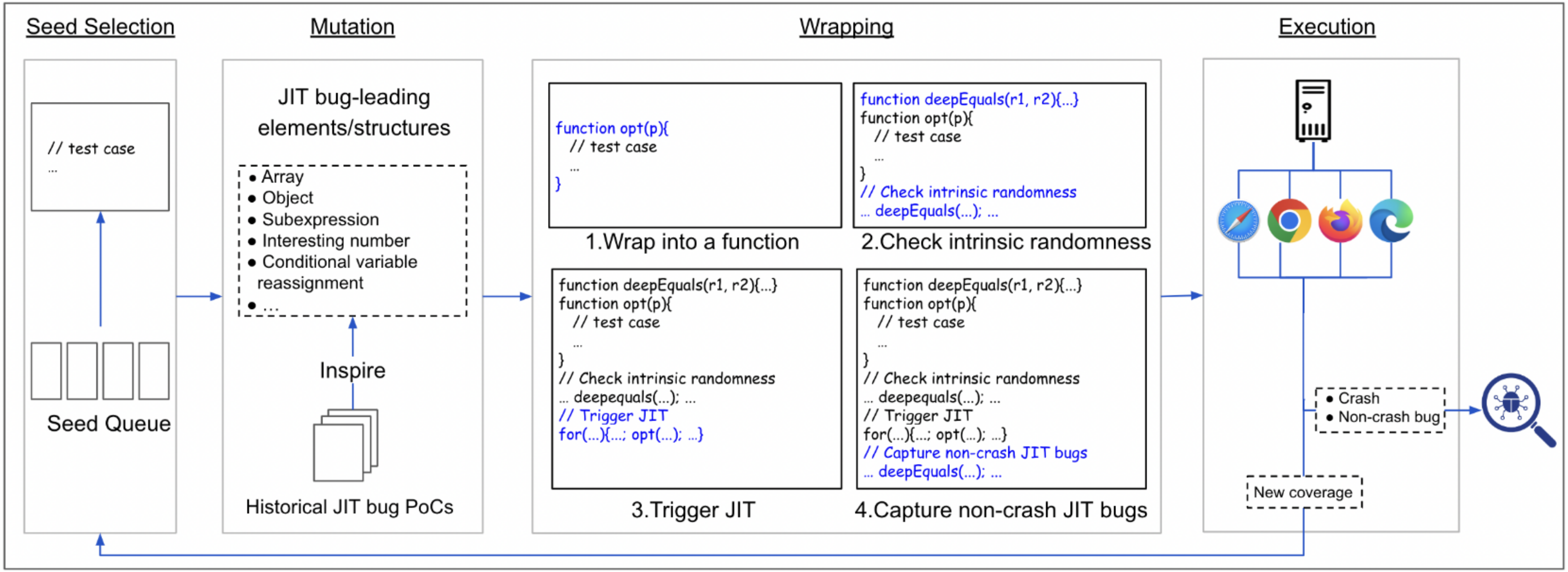
number	if (r1 === 0) Object.is(r1, r2)
	else if (isNaN(r1)) isNaN(r2)
	else r1 === r2

object (Number)	deepEquals(r1.valueOf(), r2.valueOf())

object (Date, String, RegExp, Error, Boolean)	classOf(r1) === classOf(r2)
	r1.toString() === r2.toString()

object (Array, Map, WeakMap, Set, WeakSet, JSON, Object)	classOf(r1) === classOf(r2)
	pros1 = Object.keys(r1).sort()
	pros2 = Object.keys(r2).sort()
	pros1.length === pros2.length
	for(var i = 0; i < pros1.length; i++)
	deepEquals(r1[pros1[i]], r2[pros2[i]])

FuzzJIT：workflow



```
// test case
...
```

1.Wrap into a function

```
function opt(p){
  // test case
  ...
}
```

2.Check intrinsic randomness

```
function deepEquals(r1, r2){...}
function opt(p){
  // test case
  ...
}
// Check intrinsic randomness
... deepEquals(...); ...
```

3.Trigger JIT

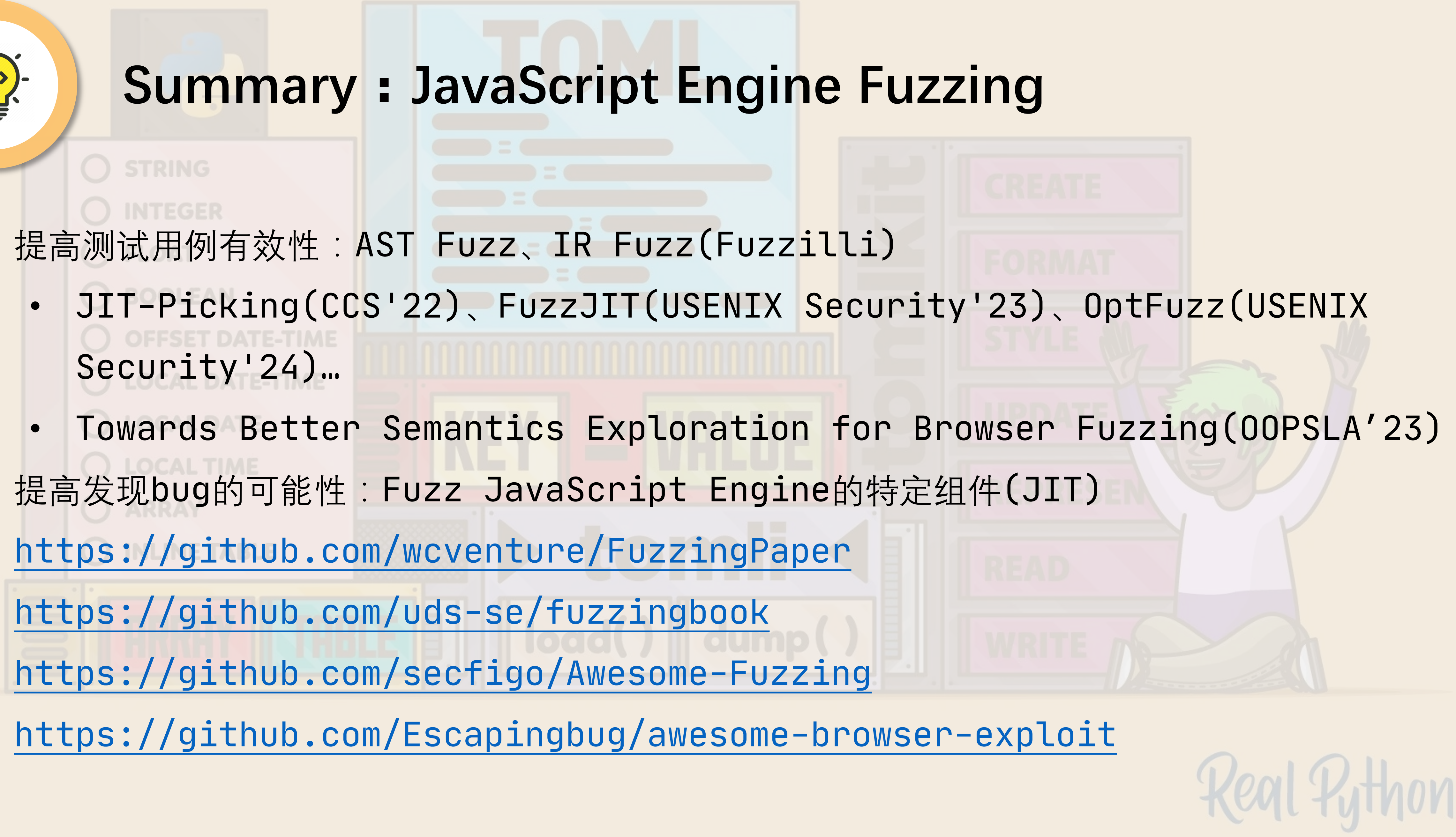
```
function deepEquals(r1, r2){...}
function opt(p){
  // test case
  ...
}
// Check intrinsic randomness
... deepequals(...); ...
// Trigger JIT
for(...){...; opt(...); ...}
```

4.Capture non-crash JIT bugs

```
function deepEquals(r1, r2){...}
function opt(p){
  // test case
  ...
}
// Check intrinsic randomness
... deepequals(...); ...
// Trigger JIT
for(...){...; opt(...); ...}
// Capture non-crash JIT bugs
... deepEquals(...); ...
```

Figure 5: The workflow of FuzzJIT.

Summary：JavaScript Engine Fuzzing



- 提高测试用例有效性：AST Fuzz、IR Fuzz(Fuzzilli)
  - JIT-Picking(CCS'22)、FuzzJIT(USENIX Security'23)、OptFuzz(USENIX Security'24)…
  - Towards Better Semantics Exploration for Browser Fuzzing(OOPSLA’23)
- 提高发现bug的可能性：Fuzz JavaScript Engine的特定组件(JIT)
- https://github.com/wcventure/FuzzingPaper
- https://github.com/uds-se/fuzzingbook
- https://github.com/secfigo/Awesome-Fuzzing
- https://github.com/Escapingbug/awesome-browser-exploit

Thanks for watching