Technische Universität Braunschweig

TESTABLE

CASA
CYBER SECURITY IN THE AGE OF LARGE-SCALE ADVERSARIES

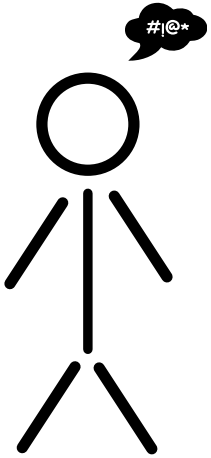
IAS | INSTITUTE FOR APPLICATION SECURITY

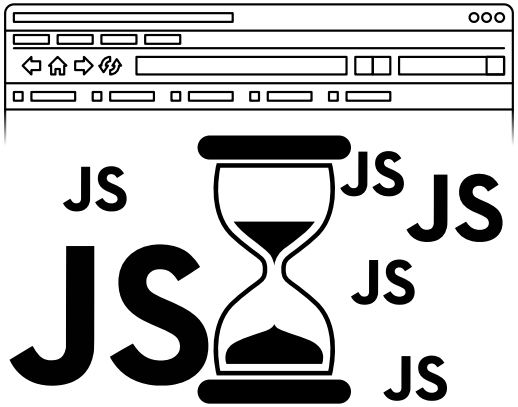
Fuzzilli: Fuzzing for JavaScript JIT Compiler Vulnerabilities

Samuel Groß*, Simon Koch+, Lukas Bernhard$, Thorsten Holz#, Martin Johns+

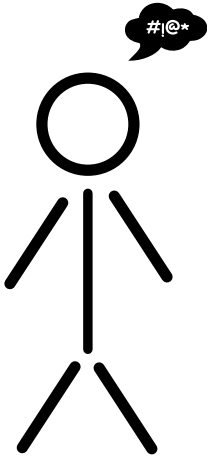
*Google, +Institute for Application Security (TU Braunschweig), $Ruhr Universität Bochum, #CISPA Helmholtz Center for Information Security

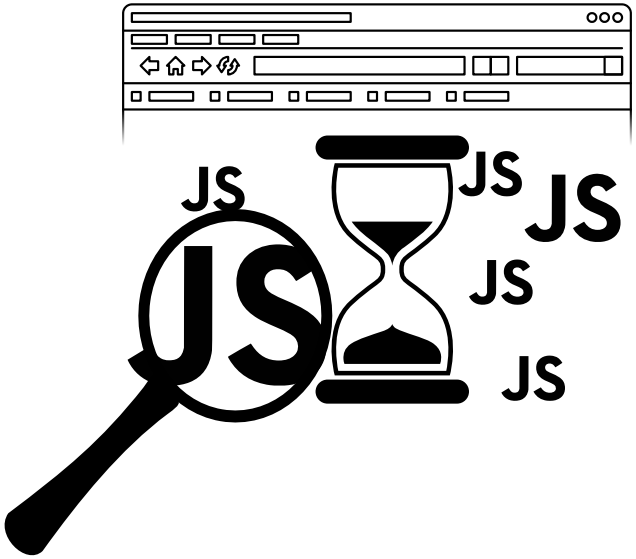
JIT Compilation Optimizes JS Execution





JIT Compilation Optimizes JS Execution

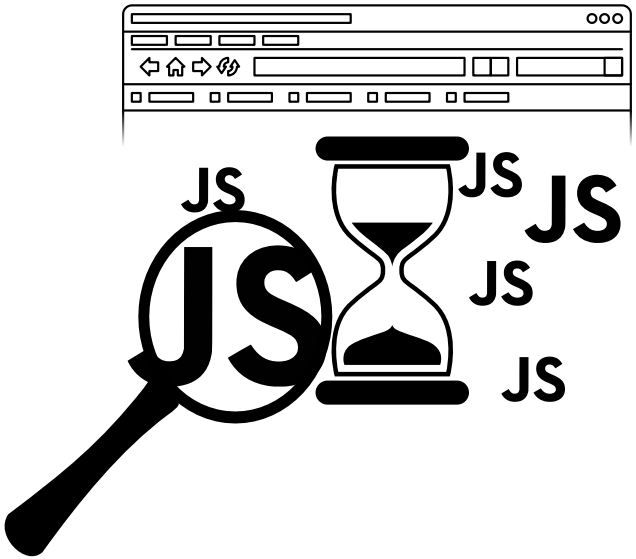




JIT Compilation Optimizes JS Execution

```
counter = 10;
while(true) {
  add(counter,10);
}

function add(a,b) {
  a += b;
}
```

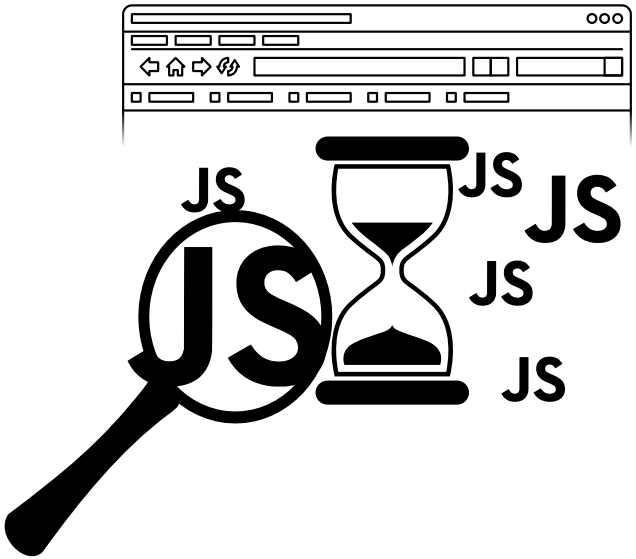


JIT Compilation Optimizes JS Execution

always int

```
counter = 10;
while(true) {
  add(counter,10);
}

function add(a,b) {
  a += b;
}
```

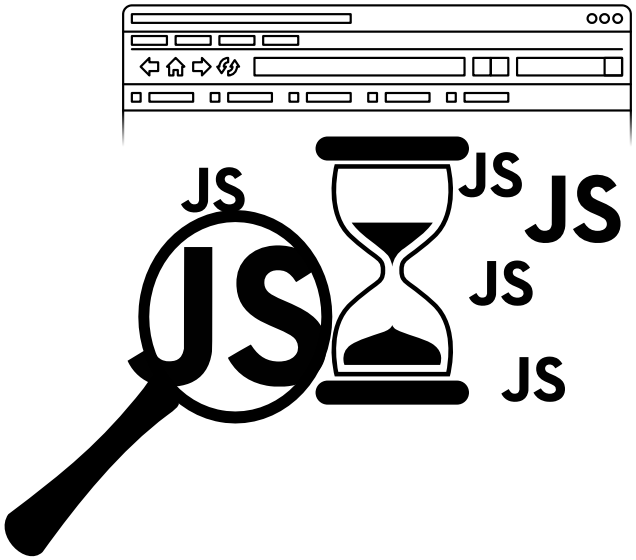


JIT Compilation Optimizes JS Execution

```
counter = 10;
while(true) {
  _add_int64(counter,10);
}

function _add_int64(a,b) {
  assert_int64(a);
  assert_int64(b);
  a += b;
}
```

speculating for int



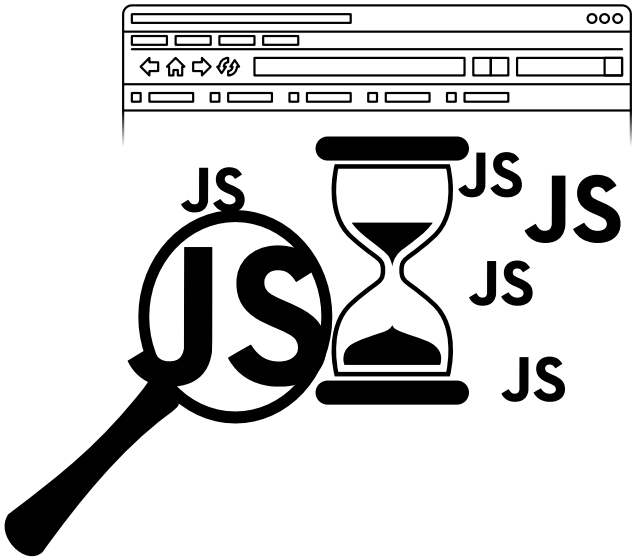
JIT Compilation Optimizes JS Execution

speculating for int

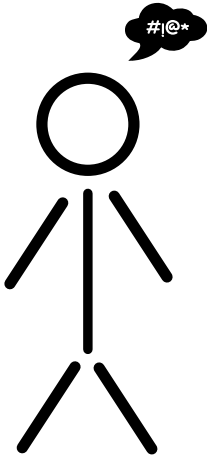
```
counter = 10;
while(true) {
  _add_int64(counter,10);
}
```

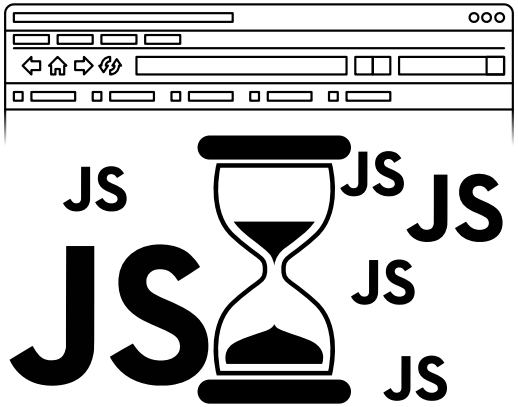
type guard

```
function _add_int64(a,b) {
  assert_int64(a);
  assert_int64(b);
  a += b;
}
```

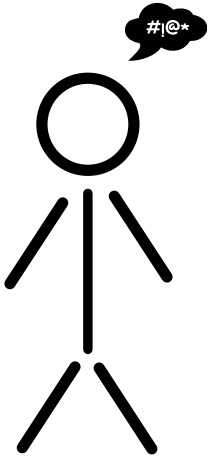


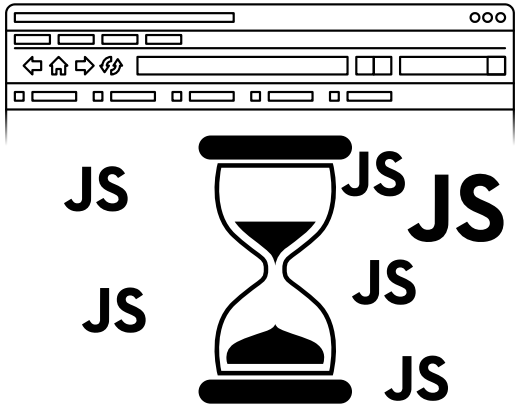
JIT Compilation Optimizes JS Execution



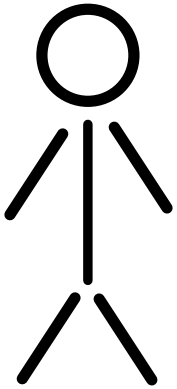


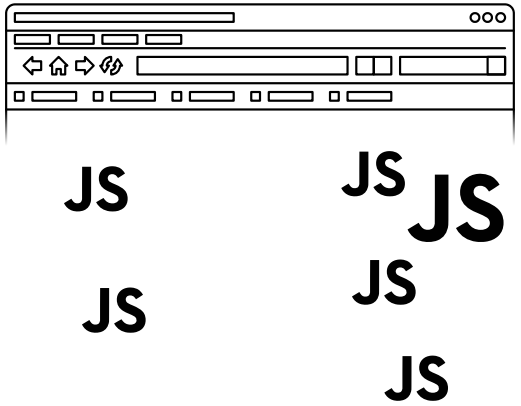
JIT Compilation Optimizes JS Execution





JIT Compilation Optimizes JS Execution





Complexity Breeds Errors

```
counter = 10;
while(true) {
  add(counter,10);
}

function add(a,b) {
  a += b;
}
```

```
counter = 10;
while(true) {
  _add_int64(counter,10);
}

function _add_int64(a,b) {
  assert_int64(a);
  assert_int64(b);
  a += b;
}
```

Complexity Breeds Errors

```
counter = 10;
while(true) {
  add(counter,10);
}

function add(a,b) {
+ do_stuff(); //no side effect
  a += b;
}
```

```
counter = 10;
while(true) {
  _add_int64(counter,10);
}

function _add_int64(a,b) {
  assert_int64(a);
  assert_int64(b);
  a += b;
}
```

Complexity Breeds Errors

```
counter = 10;
while(true) {
  add(counter,10);
}

function add(a,b) {
  do_stuff(); //no side effect
  a += b;
}
```

```
counter = 10;
while(true) {
  _add_int64(counter,10);
}

function _add_int64(a,b) {
  assert_int64(a);
  assert_int64(b);
+ do_stuff();
  a += b;
}
```

Complexity Breeds Errors

```
counter = 10;
while(true) {
  add(counter,10);
}

function add(a,b) {
  do_stuff(); //changes a to ptr
  a += b;
}
```

```
counter = 10;
while(true) {
  _add_int64(counter,10);
}

function _add_int64(a,b) {
  assert_int64(a);
  assert_int64(b);
  do_stuff();
  a += b;
}
```

Complexity Breeds Errors

```
counter = 10;
while(true) {
  add(counter,10);
}

function add(a,b) {
  do_stuff(); //changes a to ptr
  a += b;
}
```

```
counter = 10;
while(true) {
  _add_int64(counter,10);
}

function _add_int64(a,b) {
  assert_int64(a);
  assert_int64(b);
  do_stuff(); // changes a to ptr
  a += b;
}
```

Complexity Breeds Errors

```
counter = 10;
while(true) {
  add(counter,10);
}

function add(a,b) {
  do_stuff(); //changes a to ptr
  a += b;
}
```

```
counter = 10;
while(true) {
  _add_int64(counter,10);
}

function _add_int64(a,b) {
  assert_int64(a);
  assert_int64(b);
  do_stuff(); // changes a to ptr
  a += b;
}
```

JIT Compilation Recap

JIT Compilation Recap

- JIT compilation is a non trivial optimization process

JIT Compilation Recap

- JIT compilation is a non trivial optimization process

- Optimizes based on observed usage pattern

JIT Compilation Recap

- JIT compilation is a non trivial optimization process
- Optimizes based on observed usage pattern
- Complexity breeds errors

JIT Compilation Recap

- JIT compilation is a non trivial optimization process
- Optimizes based on observed usage pattern
- Complexity breeds errors
- Errors can introduce vulnerabilities

Testing JIT Compilation has Requirements

Testing JIT Compilation has Requirements

- Optimizes based on observed usage pattern

Testing JIT Compilation has Requirements

- Optimizes based on observed usage pattern

  → code needs valid syntax

Testing JIT Compilation has Requirements

- Optimizes based on observed usage pattern

  → code needs valid syntax

  → code needs to be semantically correct

Testing JIT Compilation has Requirements

- Optimizes based on observed usage pattern

  → code needs valid syntax

  → code needs to be semantically correct

  → we need a lot of different samples trying the most improbable combinations

Testing JIT Compilation has Requirements

- Optimizes based on observed usage pattern

  → code needs valid syntax

  → code needs to be semantically correct

  → we need a lot of different samples trying the most improbable combinations

- Solution: Fuzzilli is a fuzzer using an IL to generate valid JavaScript code

Fuzzilli is an IL Based JS Fuzzer

Fuzzilli is an IL Based JS Fuzzer

```
v0 <- BeginPlainFunction -> v1
    v2 <- CreateArray [v1, v1, v1]
    v3 <- LoadInt ‘1’
    v4 <- CallMethod ‘slice’, v2, [v3]
    Return v4
EndPlainFunction
v5 <- LoadFloat ‘13.37’
V6 <- CallFunction v0, [v5]
```

Fuzzilli is an IL Based JS Fuzzer

```
v0 <- BeginPlainFunction -> v1
    v2 <- CreateArray [v1, v1, v1]
    v3 <- LoadInt ‘1’
    v4 <- CallMethod ‘slice’, v2, [v3]
    Return v4
EndPlainFunction
v5 <- LoadFloat ‘13.37’
V6 <- CallFunction v0, [v5]
```

```
function v0(v1) {
    const v2 = [v1, v1, v1];
    const v4 = v2.slice(1);
    return v4;
}
const v6 = v0(13.37);
```

Mutations are Performed on the IL

```
v0 <- BeginPlainFunction -> v1
    v2 <- CreateArray [v1, v1, v1]
    v3 <- LoadInt ‘1’
    v4 <- CallMethod ‘slice’, v2, [v3]
    Return v4
EndPlainFunction
v5 <- LoadFloat ‘13.37’
V6 <- CallFunction v0, [v5]
```

```
function v0(v1) {
    const v2 = [v1, v1, v1];
    const v4 = v2.slice(1);
    return v4;
}
const v6 = v0(13.37);
```

Mutations are Performed on the IL

```
v0 <- BeginPlainFunction -> v1
    v2 <- CreateArray [v1, v1, v1]
    v3 <- LoadInt '1'
    v4 <- CallMethod 'fill', v2, [v3]
    Return v4
EndPlainFunction
v5 <- LoadFloat '13.37'
V6 <- CallFunction v0, [v5]
```

```
function v0(v1) {
    const v2 = [v1, v1, v1];
    const v4 = v2.slice(1);
    return v4;
}
const v6 = v0(13.37);
```

Mutations are Performed on the IL

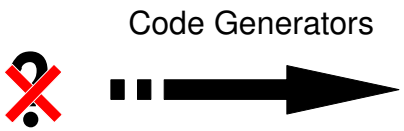
```
v0 <- BeginPlainFunction -> v1
    v2 <- CreateArray [v1, v1, v1]
    v3 <- LoadInt '1'
    v4 <- CallMethod 'fill', v2, [v3]
    Return v4
EndPlainFunction
v5 <- LoadFloat '13.37'
V6 <- CallFunction v0, [v5]
```

```
function v0(v1) {
    const v2 = [v1, v1, v1];
    const v4 = v2.fill(1);
    return v4;
}
const v6 = v0(13.37);
```

No Seed Corpus Required


```
v0 <- BeginPlainFunction -> v1
    v2 <- CreateArray [v1, v1, v1]
    v3 <- LoadInt ‘1’
    v4 <- CallMethod ‘fill’, v2, [v3]
    Return v4
EndPlainFunction
v5 <- LoadFloat ‘13.37’
V6 <- CallFunction v0, [v5]
```

No Seed Corpus Required




```
v0 <- BeginPlainFunction -> v1
    v2 <- CreateArray [v1, v1, v1]
    v3 <- LoadInt ‘1’
    v4 <- CallMethod ‘fill’, v2, [v3]
    Return v4
EndPlainFunction
v5 <- LoadFloat ‘13.37’
V6 <- CallFunction v0, [v5]
```

Assessing Fuzzilli

Assessing Fuzzilli

- Quantitative assessment

  → what code coverage does Fuzzilli reach?

Assessing Fuzzilli

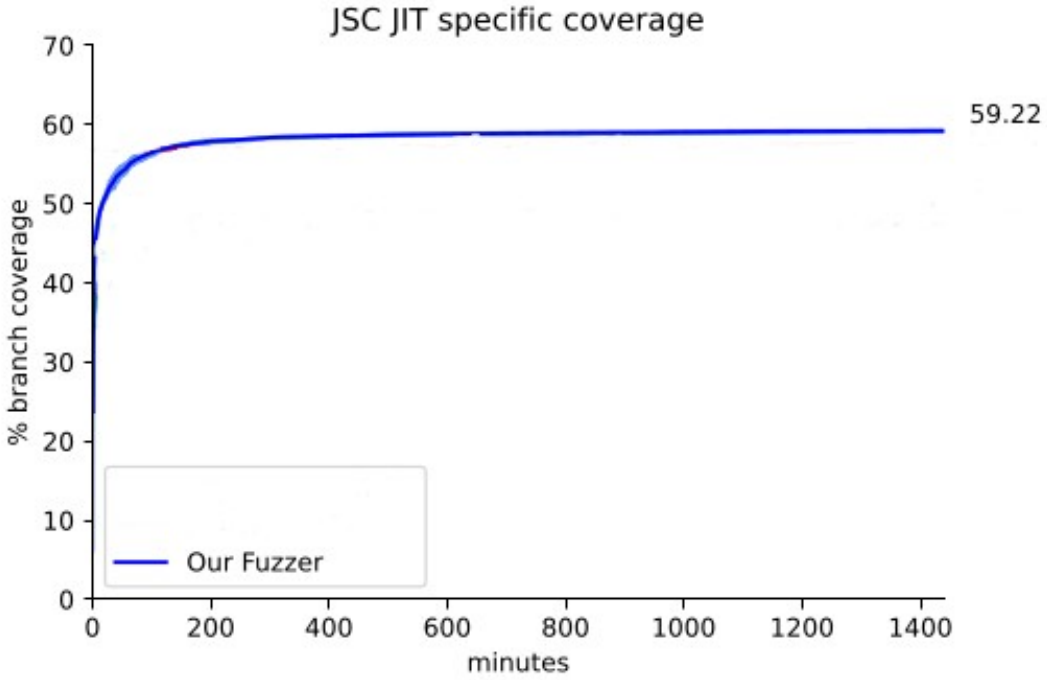
- Quantitative assessment

  → what code coverage does Fuzzilli reach?

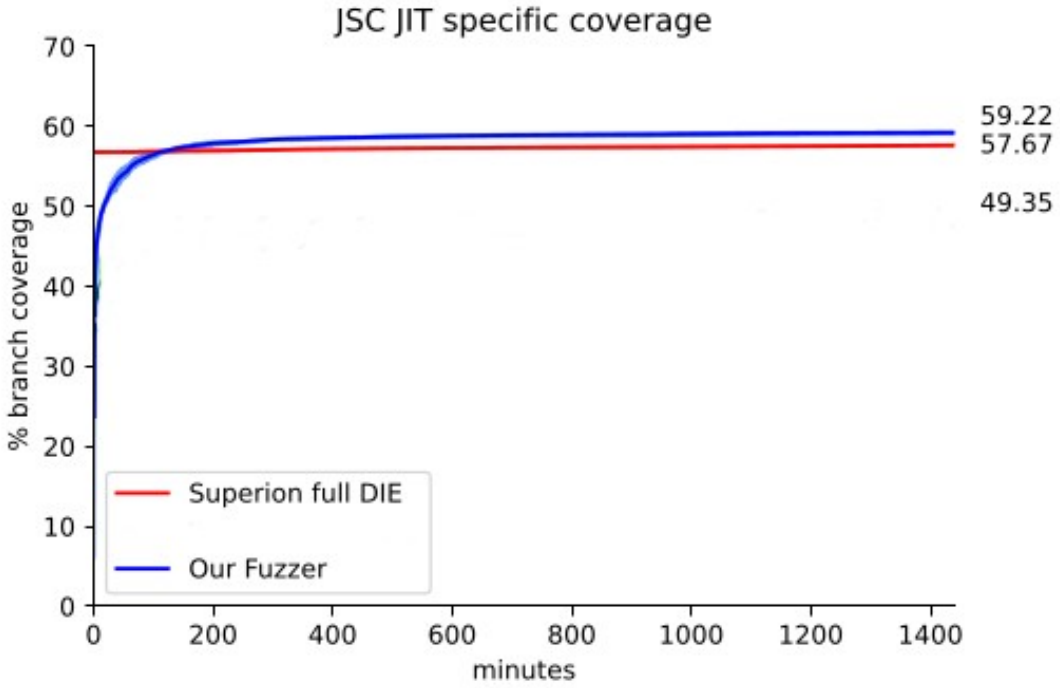
- Qualitative assessment

  → do we find bugs that were previously missed?

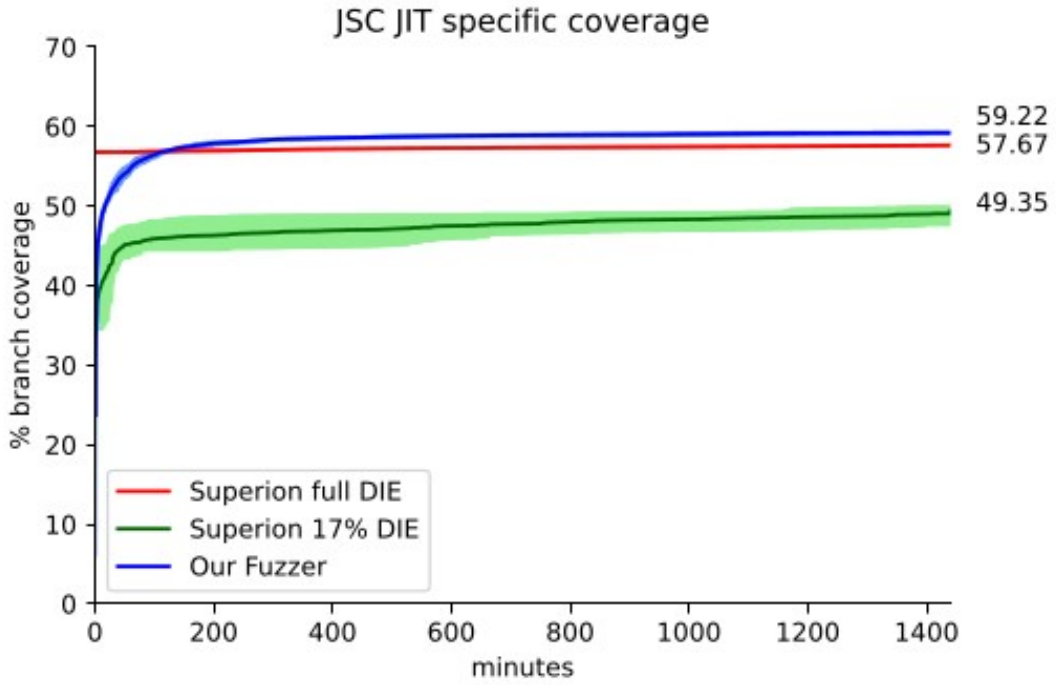
Fuzzilli Has Good JIT Compiler Code Coverage



Fuzzilli Has Good JIT Compiler Code Coverage



Fuzzilli Has Good JIT Compiler Code Coverage



Fuzzilli Finds Bugs

Engine	Issue	Type	Runtime
SM	CVE-2019-9813	Type Safety	✓
SM	CVE-2019-9791	Type Safety	✓
SM	CVE-2019-11707	Type Safety	✓
V8	Issue 944062	Type Safety	✓
V8	Issue 944865	Type Safety	✓
JSC	CVE-2019-8671	Type Safety	✓
JSC	CVE-2019-8765	Type Safety	✘
V8	Issue 939316	Spatial Safety	✘
JSC	CVE-2019-8518	Spatial Safety	✓
JSC	CVE-2019-8622	Temporal Safety	✓
JSC	CVE-2019-8672	Temporal Safety	✓
JSC	CVE-2019-8558	Temporal Safety	✘
JSC	CVE-2019-8623	Uninit Data	✓
JSC	CVE-2019-8611	Uninit Data	✓
SM	CVE-2019-9792	Misc	✓
SM	CVE-2019-9816	Misc	✓
V8	Issue 958717	Misc	✓

Fuzzilli Finds Previously Missed Bugs

Engine	Issue	Type	Runtime	Age
SM	CVE-2019-9813	Type Safety	✓	18
SM	CVE-2019-9791	Type Safety	✓	> 36
SM	CVE-2019-11707	Type Safety	✓	20
V8	Issue 944062	Type Safety	✓	1
V8	Issue 944865	Type Safety	✓	8
JSC	CVE-2019-8671	Type Safety	✓	9
JSC	CVE-2019-8765	Type Safety	✘	> 6
V8	Issue 939316	Spatial Safety	✘	4
JSC	CVE-2019-8518	Spatial Safety	✓	9
JSC	CVE-2019-8622	Temporal Safety	✓	> 36
JSC	CVE-2019-8672	Temporal Safety	✓	30
JSC	CVE-2019-8558	Temporal Safety	✘	> 30
JSC	CVE-2019-8623	Uninit Data	✓	24
JSC	CVE-2019-8611	Uninit Data	✓	4
SM	CVE-2019-9792	Misc	✓	4
SM	CVE-2019-9816	Misc	✓	> 36
V8	Issue 958717	Misc	✓	4

Fuzzilli Finds Previously Missed Bugs

Engine	Issue	Type	Runtime	Age
SM	CVE-2019-9813	Type Safety	✓	18
SM	CVE-2019-9791	Type Safety	✓	> 36
SM	CVE-2019-11707	Type Safety	✓	20
V8	Issue 944062	Type Safety	✓	1
V8	Issue 944865	Type Safety	✓	8
JSC	CVE-2019-8671	Type Safety	✓	9
JSC	CVE-2019-8765	Type Safety	✘	> 6
V8	Issue 939316	Spatial Safety	✘	4
JSC	CVE-2019-8518	Spatial Safety	✓	9
JSC	CVE-2019-8622	Temporal Safety	✓	> 36
JSC	CVE-2019-8672	Temporal Safety	✓	30
JSC	CVE-2019-8558	Temporal Safety	✘	> 30
JSC	CVE-2019-8623	Uninit Data	✓	24
JSC	CVE-2019-8611	Uninit Data	✓	4
SM	CVE-2019-9792	Misc	✓	4
SM	CVE-2019-9816	Misc	✓	> 36
V8	Issue 958717	Misc	✓	4

Fuzzilli Finds Previously Missed Bugs

Engine	Issue	Type	Runtime	Age
SM	CVE-2019-9813	Type Safety	✓	18
SM	CVE-2019-9791	Type Safety	✓	> 36
SM	CVE-2019-11707	Type Safety	✓	20
V8	Issue 944062	Type Safety	✓	1
V8	Issue 944865	Type Safety	✓	8
JSC	CVE-2019-8671	Type Safety	✓	9
JSC	CVE-2019-8765	Type Safety	✘	> 6
V8	Issue 939316	Spatial Safety	✘	4
JSC	CVE-2019-8518	Spatial Safety	✓	9
JSC	CVE-2019-8622	Temporal Safety	✓	> 36
JSC	CVE-2019-8672	Temporal Safety	✓	30
JSC	CVE-2019-8558	Temporal Safety	✘	> 30
JSC	CVE-2019-8623	Uninit Data	✓	24
JSC	CVE-2019-8611	Uninit Data	✓	4
SM	CVE-2019-9792	Misc	✓	4
SM	CVE-2019-9816	Misc	✓	> 36
V8	Issue 958717	Misc	✓	4

Fuzzilli Finds Previously Missed Bugs

Engine	Issue	Type	Runtime	Age
SM	CVE-2019-9813	Type Safety	✓	18
SM	CVE-2019-9791	Type Safety	✓	> 36
SM	CVE-2019-11707	Type Safety	✓	20
V8	Issue 944062	Type Safety	✓	1
V8	Issue 944865	Type Safety	✓	8
JSC	CVE-2019-8671	Type Safety	✓	9
JSC	CVE-2019-8765	Type Safety	✘	> 6
V8	Issue 939316	Spatial Safety	✘	4
JSC	CVE-2019-8518	Spatial Safety	✓	9
JSC	CVE-2019-8622	Temporal Safety	✓	> 36
JSC	CVE-2019-8672	Temporal Safety	✓	30
JSC	CVE-2019-8558	Temporal Safety	✘	> 30
JSC	CVE-2019-8623	Uninit Data	✓	24
JSC	CVE-2019-8611	Uninit Data	✓	4
SM	CVE-2019-9792	Misc	✓	4
SM	CVE-2019-9816	Misc	✓	> 36
V8	Issue 958717	Misc	✓	4

Fuzzilli Finds Previously Missed Bugs

Engine	Issue	Type	Runtime	Age
SM	CVE-2019-9813	Type Safety	✓	18
SM	CVE-2019-9791	Type Safety	✓	> 36
SM	CVE-2019-11707	Type Safety	✓	20
V8	Issue 944062	Type Safety	✓	1
V8	Issue 944865	Type Safety	✓	8
JSC	CVE-2019-8671	Type Safety	✓	9
JSC	CVE-2019-8765	Type Safety	✘	> 6
V8	Issue 939316	Spatial Safety	✘	4
JSC	CVE-2019-8518	Spatial Safety	✓	9
JSC	CVE-2019-8622	Temporal Safety	✓	> 36
JSC	CVE-2019-8672	Temporal Safety	✓	30
JSC	CVE-2019-8558	Temporal Safety	✘	> 30
JSC	CVE-2019-8623	Uninit Data	✓	24
JSC	CVE-2019-8611	Uninit Data	✓	4
SM	CVE-2019-9792	Misc	✓	4
SM	CVE-2019-9816	Misc	✓	> 36
V8	Issue 958717	Misc	✓	4

Fuzzilli Finds Previously Missed Bugs

Engine	Issue	Type	Runtime	Age
SM	CVE-2019-9813	Type Safety	✓	18
SM	CVE-2019-9791	Type Safety	✓	> 36
SM	CVE-2019-11707	Type Safety	✓	20
V8	Issue 944062	Type Safety	✓	1
V8	Issue 944865	Type Safety	✓	8
JSC	CVE-2019-8671	Type Safety	✓	9
JSC	CVE-2019-8765	Type Safety	✘	> 6
V8	Issue 939316	Spatial Safety	✘	4
JSC	CVE-2019-8518	Spatial Safety	✓	9
JSC	CVE-2019-8622	Temporal Safety	✓	> 36
JSC	CVE-2019-8672	Temporal Safety	✓	30
JSC	CVE-2019-8558	Temporal Safety	✘	> 30
JSC	CVE-2019-8623	Uninit Data	✓	24
JSC	CVE-2019-8611	Uninit Data	✓	4
SM	CVE-2019-9792	Misc	✓	4
SM	CVE-2019-9816	Misc	✓	> 36
V8	Issue 958717	Misc	✓	4

Summary

Summary

Fuzzilli Finds Previously Missed Bugs

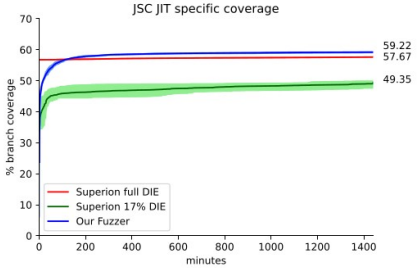
Engine	Issue	Type	Runtime	Age
SM	CVE-2019-9813	Type Safety	✓	18
SM	CVE-2019-9791	Type Safety	✓	> 36
SM	CVE-2019-11707	Type Safety	✓	20
V8	Issue 944062	Type Safety	✓	1
V8	Issue 944865	Type Safety	✓	8
JSC	CVE-2019-8671	Type Safety	✓	9
JSC	CVE-2019-8765	Type Safety	✘	> 6
V8	Issue 939316	Spatial Safety	✘	4
JSC	CVE-2019-8518	Spatial Safety	✓	9
JSC	CVE-2019-8622	Temporal Safety	✓	> 36
JSC	CVE-2019-8672	Temporal Safety	✓	30
JSC	CVE-2019-8558	Temporal Safety	✘	> 30
JSC	CVE-2019-8623	Uninit Data	✓	24
JSC	CVE-2019-8611	Uninit Data	✓	4
SM	CVE-2019-9792	Misc	✓	4
SM	CVE-2019-9816	Misc	✓	> 36
V8	Issue 958717	Misc	✓	4

Summary

Fuzzilli Finds Previously Missed Bugs

Engine	Issue	Type	Runtime	Age
SM	CVE-2019-9813	Type Safety	✓	18
SM	CVE-2019-9791	Type Safety	✓	> 36
SM	CVE-2019-11707	Type Safety	✓	20
V8	Issue 944062	Type Safety	✓	1
V8	Issue 944865	Type Safety	✓	8
JSC	CVE-2019-8671	Type Safety	✓	9
JSC	CVE-2019-8765	Type Safety	✘	> 6
V8	Issue 939316	Spatial Safety	✘	4
JSC	CVE-2019-8518	Spatial Safety	✓	9
JSC	CVE-2019-8622	Temporal Safety	✓	> 36
JSC	CVE-2019-8672	Temporal Safety	✓	30
JSC	CVE-2019-8558	Temporal Safety	✘	> 30
JSC	CVE-2019-8623	Uninit Data	✓	24
JSC	CVE-2019-8611	Uninit Data	✓	4
SM	CVE-2019-9792	Misc	✓	4
SM	CVE-2019-9816	Misc	✓	> 36
V8	Issue 958717	Misc	✓	4

Fuzzilli Has Good JIT Code Coverage

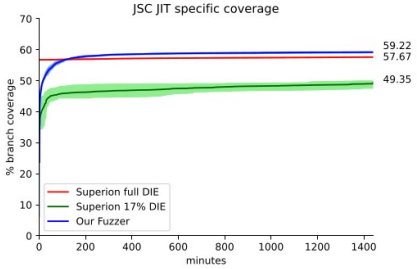


Summary

Fuzzilli Finds Previously Missed Bugs

Engine	Issue	Type	Runtime	Age
SM	CVE-2019-9813	Type Safety	✓	18
SM	CVE-2019-9791	Type Safety	✓	> 36
SM	CVE-2019-11707	Type Safety	✓	20
V8	Issue 944062	Type Safety	✓	1
V8	Issue 944865	Type Safety	✓	8
JSC	CVE-2019-8671	Type Safety	✓	9
JSC	CVE-2019-8765	Type Safety	✘	> 6
V8	Issue 939316	Spatial Safety	✘	4
JSC	CVE-2019-8518	Spatial Safety	✓	9
JSC	CVE-2019-8622	Temporal Safety	✓	> 36
JSC	CVE-2019-8672	Temporal Safety	✓	30
JSC	CVE-2019-8558	Temporal Safety	✘	> 30
JSC	CVE-2019-8623	Uninit Data	✓	24
JSC	CVE-2019-8611	Uninit Data	✓	4
SM	CVE-2019-9792	Misc	✓	4
SM	CVE-2019-9816	Misc	✓	> 36
V8	Issue 958717	Misc	✓	4

Fuzzilli Has Good JIT Code Coverage



Questions?

Email: simon.koch@tu-braunschweig.de

This Research was funded by the
Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy - EXC 2092 CASA – 390781972
German Federal Ministry of Education and Research, project KMU-Fuzz – 16KIS1523
European Union's Horizon 2020 research and innovation program under grant agreement No 101019206

Fuzzilli is Still Finding Bugs with Security Implications

WebKit/JavaScriptCore

- Issue 185328: DFG Compiler uses incorrect output register for NumberIsInteger operation
- CVE-2018-4299: performProxyCall leaks internal object to script
- CVE-2018-4359: compileMathIC produces incorrect machine code
- CVE-2019-8518: OOB access in FTL JIT due to LICM moving array access before the bounds check
- CVE-2019-8558: CodeBlock UaF due to dangling Watchpoints
- CVE-2019-8611: AIR optimization incorrectly removes assignment to register
- CVE-2019-8623: Loop-invariant code motion (LICM) in DFG JIT leaves stack variable uninitialized
- CVE-2019-8622: DFG's doesGC() is incorrect about the HasIndexedProperty operation's behaviour on StringObjects
- CVE-2019-8671: DFG: Loop-invariant code motion (LICM) leaves object property access unguarded
- CVE-2019-8672: JSValue use-after-free in ValueProfiles
- CVE-2019-8678: JSC fails to run haveABadTime() when some prototypes are modified, leading to type confusions
- CVE-2019-8685: JSPropertyNameEnumerator uses wrong structure IDs
- CVE-2019-8765: GetterSetter type confusion during DFG compilation
- CVE-2019-8820: Type confusion during bailout when reconstructing arguments objects
- CVE-2019-8844: ObjectAllocationSinkingPhase shouldn't insert hints for allocations which are no longer valid
- CVE-2020-3901: GetterSetter type confusion in FTL JIT code (due to not always safe LICM)
- CVE-2021-30851: Missing lock during concurrent HashTable lookup
- CVE-2021-30818: Type confusion when reconstructing arguments on DFG OSR Exit
- CVE-2022-46696: Assertion failure due to missing exception check in JIT-compiled code
- CVE-2022-46699: Assertion failure due to incorrect caching of special properties in ICs
- CVE-2022-46700: Intl.Locale.prototype.hourCycles leaks empty JSValue to script

Gecko/Spidermonkey

- CVE-2018-12386: IonMonkey register allocation bug leads to type confusions
- CVE-2019-9791: IonMonkey's type inference is incorrect for constructors entered via OSR
- CVE-2019-9792: IonMonkey leaks JS_OPTIMIZED_OUT magic value to script
- CVE-2019-9816: unexpected ObjectGroup in ObjectGroupDispatch operation
- CVE-2019-9813: IonMonkey compiled code fails to update inferred property types, leading to type confusions
- CVE-2019-11707: IonMonkey incorrectly predicts return type of Array.prototype.pop, leading to type confusions
- CVE-2020-15656: Type confusion for special arguments in IonMonkey
- CVE-2022-42928: Missing KeepAlive annotations for some BigInt operations may lead to memory corruption
- CVE-2022-45406: Use-after-free of a JavaScript Realm

Chromium/v8

- Issue 939316: Turbofan may read a Map pointer out-of-bounds when optimizing Reflect.construct
- Issue 944062: JSCallReducer::ReduceArrayIndexOfIncludes fails to insert Map checks
- CVE-2019-5831: Incorrect map processing in V8
- Issue 944865: Invalid value representation in V8
- CVE-2019-5841: Bug in inlining heuristic
- CVE-2019-5847: V8 sealed/frozen elements cause crash
- CVE-2019-5853: Memory corruption in regexp length check
- Issue 992914: Map migration doesn't respect element kinds, leading to type confusion
- CVE-2020-6512: Type Confusion in V8
- CVE-2020-16006: Memory corruption due to improperly handled hash collision in DescriptorArray
- CVE-2021-37991: Race condition during concurrent JIT compilation
- Issue 1359937: Deserialization of BigInts could result in invalid -0n value
- Issue 1377775: Incorrect type check when inlining Array.prototype.at in Turbofan

Duktape

- Issue 2323: Unstable valstack pointer in putprop
- Issue 2320: Memcmp pointer overflow in string builtin

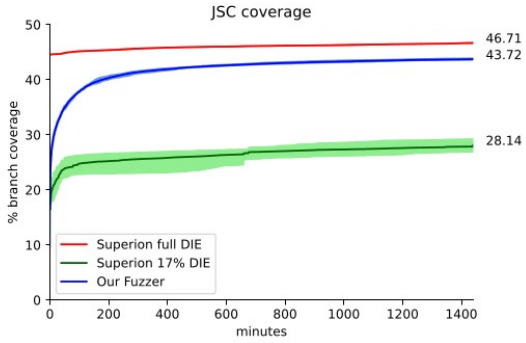
JerryScript

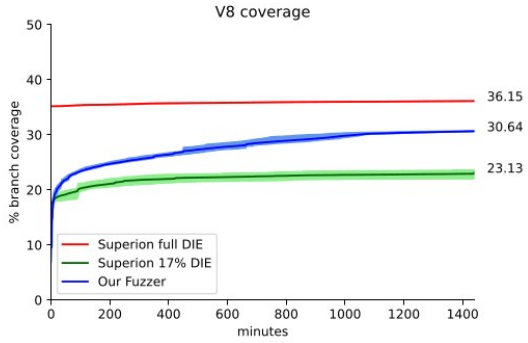
- CVE-2020-13991: Incorrect release of spread arguments
- Issue 3784: Memory corruption due to incorrect property enumeration
- CVE-2020-13623: Stack overflow via property keys for Proxy objects
- CVE-2020-13649 (1): Memory corruption due to error handling in case of OOM
- CVE-2020-13649 (2): Memory corruption due to error handling in case of OOM
- CVE-2020-13622: Memory corruption due to incorrect handling of property keys for Proxy objects
- CVE-2020-14163: Memory corruption due to race condition triggered by garbage collection when adding key/value pairs
- Issue 3813: Incorrect error handling in SerializeJSONProperty function
- Issue 3814: Unexpected Proxy object in ecma_op_function_has_instance assertion
- Issue 3836: Memory corruption due to incorrect TypedArray initialization
- Issue 3837: Memory corruption due to incorrect memory handling in getOwnPropertyDescriptor

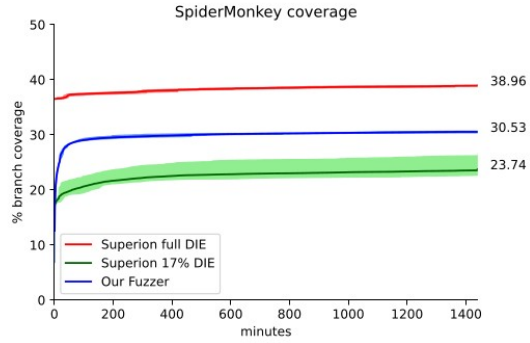
Hermes

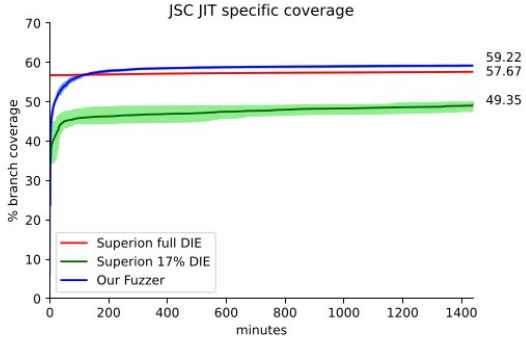
- CVE-2020-1912: Memory corruption when executing lazily compiled inner generator functions
- CVE-2020-1914: Bytecode corruption when handling the SaveGeneratorLong instruction

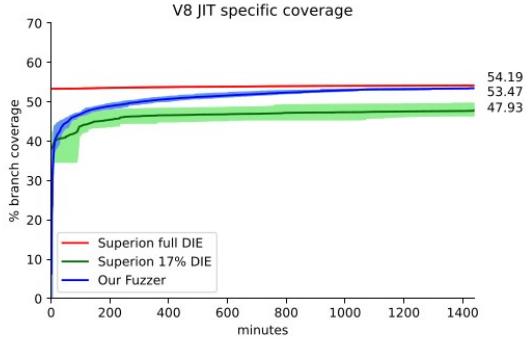
Fuzzilli Code Coverage

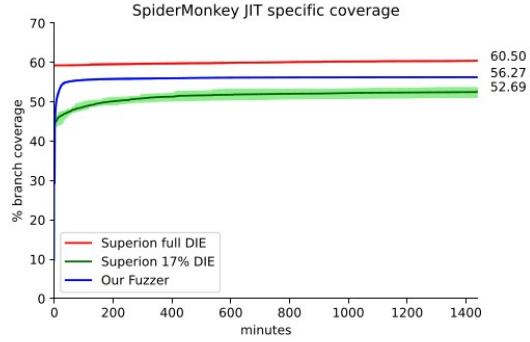












Fuzzilli Better than the DIE Seed for JIT Compiler Files

Engine	DIE Coverage	Superion Coverage	Our Coverage
JSC	44.56% / 17%	46.71% / 28.14%	43.72%
JSC JIT	56.78% / 34.98%	57.67% / 49.35%	59.22%
V8	35.17% / 17%	36.15% / 23.13%	30.64%
V8 JIT	53.33% / 38.82%	54.22% / 47.93%	53.47%
SM	36.49% / 17%	38.96% / 23.74%	30.53%
SM JIT	59.28% / 43.82%	60.50% / 52.69%	56.27%