TÉCNICO LISBOA

Presentation

Favocado: Fuzzing the Binding Code of JavaScript Engines Using Semantically Correct Test Cases

Joseph Thachil George-PhD- Student

joseph.thachil.george@tecnico.ulisboa.pt

TÉCNICO LISBOA

OVERVIEW

- Introduction
- Favocado
- Architecture and Design Of Favocado
- Favocado-Evaluation
- Comparison
- Conclusion and Observation

TÉCNICO LISBOA

INTRODUCTION

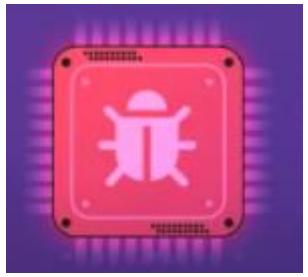
Favocado: Fuzzing the Binding Code of JavaScript Engines Using Semantically Correct Test Cases

- JavaScript Engines
- Binding Code
- Fuzzing the Binding Code
- Favocado
- Semantically Correct Test Cases

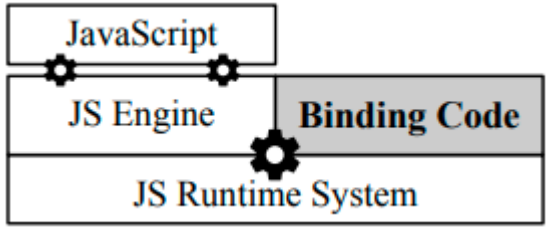
TÉCNICO LISBOA

INTRODUCTION- CONT.

Favocado: Fuzzing the Binding Code of JavaScript Engines Using Semantically Correct Test Cases



JavaScript Engines- Vulnerabilities



Binding Code-Vulnerabilities

JavaScript Fuzzers

Domato

TÉCNICO LISBOA

INTRODUCTION- CONT.

Limitation of Domato

```
>>> US_PHONE_GRAMMAR = {
>>>     "<start>": ["<phone-number>"],
>>>     "<phone-number>": ["(<area>)<exchange>-<line>"],
>>>     "<area>": ["<lead-digit><digit><digit>"],
>>>     "<exchange>": ["<lead-digit><digit><digit>"],
>>>     "<line>": ["<digit><digit><digit><digit>"],
>>>     "<lead-digit>": ["2", "3", "4", "5", "6", "7", "8", "9"],
>>>     "<digit>": ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"]
>>> }
>>>
>>> assert is_valid_grammar(US_PHONE_GRAMMAR)
```

TÉCNICO LISBOA

FAVOCADO

Why FAVOCADO ?

- Generate Syntactically and semantically Correct Test case
- Build effective binding Fuzzer

Test Case Example

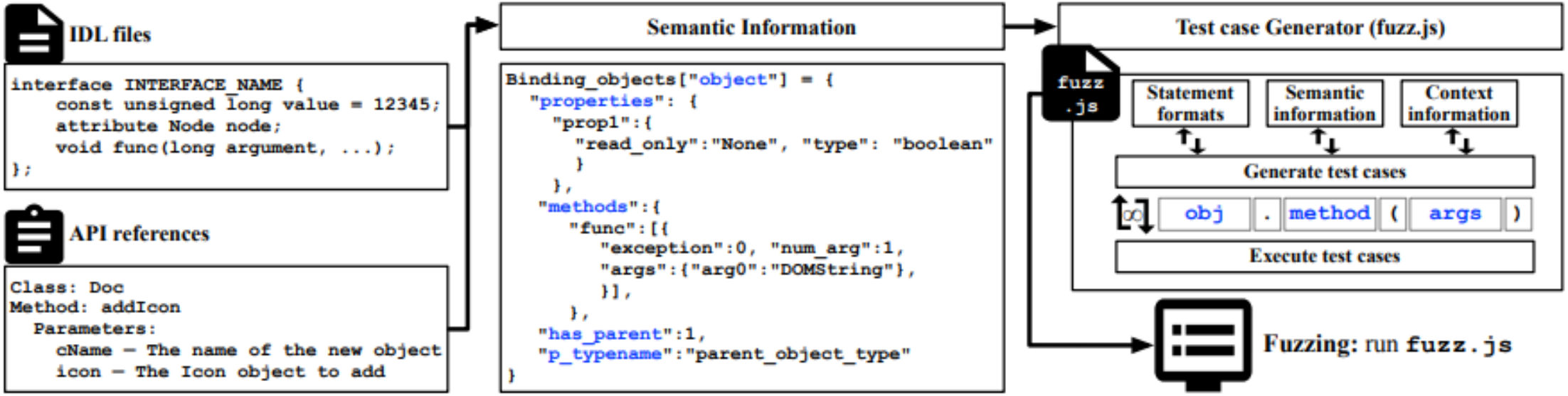
```
1 var cb = this.getField("CheckBox");
2 cb.checkThisBox(0, true);
```

TÉCNICO LISBOA

ARCHITECTURE AND DESIGN OF FAVOCADO

TÉCNICO LISBOA

ARCHITECTURE OF FAVOCADO



TÉCNICO LISBOA

FAVOCADO

Semantic Information Construction

Favocado Parses

- Binding object names
- Binding object methods
- Binding object properties.

Favocado Parses

- Related binding objects

TÉCNICO LISBOA

FAVOCADO

Semantic Information Construction

```
Binding_objects["HTMLDialogElement"] = {
  "properties":
  {
    "open":
    {
      "read_only":"None", "type":"boolean"
    },
    "returnValue":
    {
      "read_only":"None", "type":"DOMString"
    }
  },
  "methods":
  {
    "close":
    {
      "exception":0, "numarg":1,
      "args":{"arg0":"DOMString"},
    },
    "showModal":
    {
      "exception":1, "numarg":0,
      "args":{},
    },
    "show":
    {
      "exception":0, "numarg":0,
      "args":{},
    }
  },
  "has_parent":1,
  "p_typename":"HTMLElement"
}
```

+

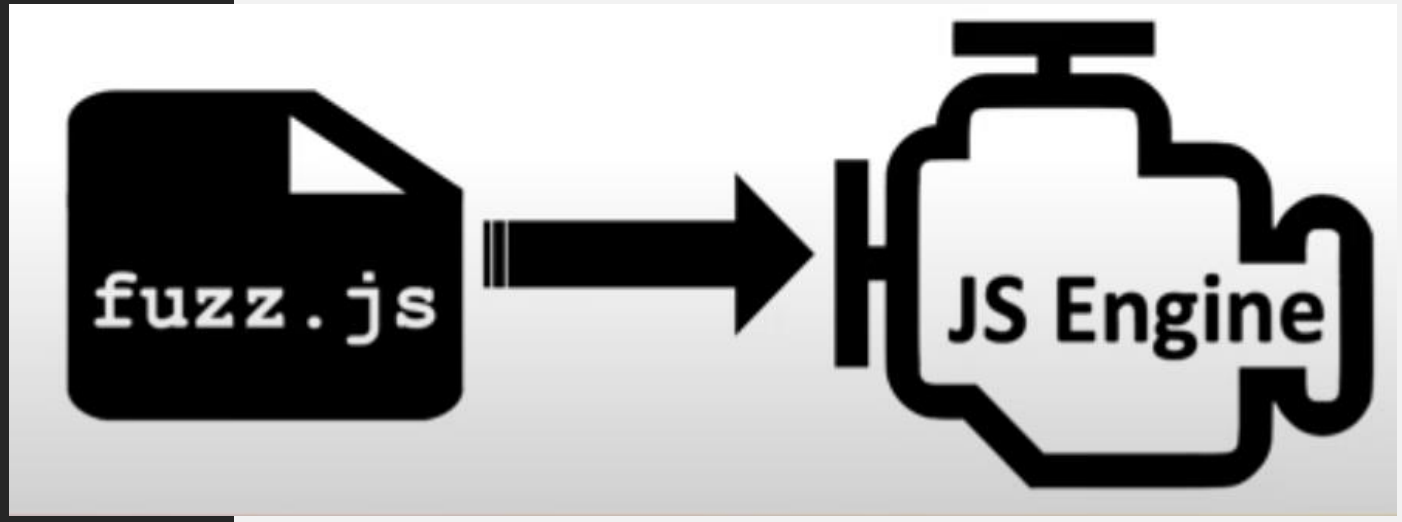
```
"ImageCapture":
[{
  "Blob", "ImageBitmap", "MediaStreamTrack", "
      PhotoCapabilities"
}]

"Crypto":
[{
  "ArrayBufferView", "SubtleCrypto"
}]
```

TÉCNICO LISBOA

FAVOCADO- DYNAMIC TEST CASE GENERATOR AND FUZZING

- Select Binding objects
- Generate a test case generator (fuzz.js) with semantic info of the selected binding object
- Executing the text case generator.



TÉCNICO LISBOA

FAVOCADO-TEST CASE GENERATOR

fuzz.js

```
1  Initialize all objects

2  while (1) {
3     Select a statement format
4     Complete the selected format
5     Log the complete statement
6     try {
7        Execute the statement
8     } catch (error) {
9        Continue the loop
10    }
11 }
```

- Statement Formats
- Semantic Information of Binding Code
- Context Information

TÉCNICO LISBOA

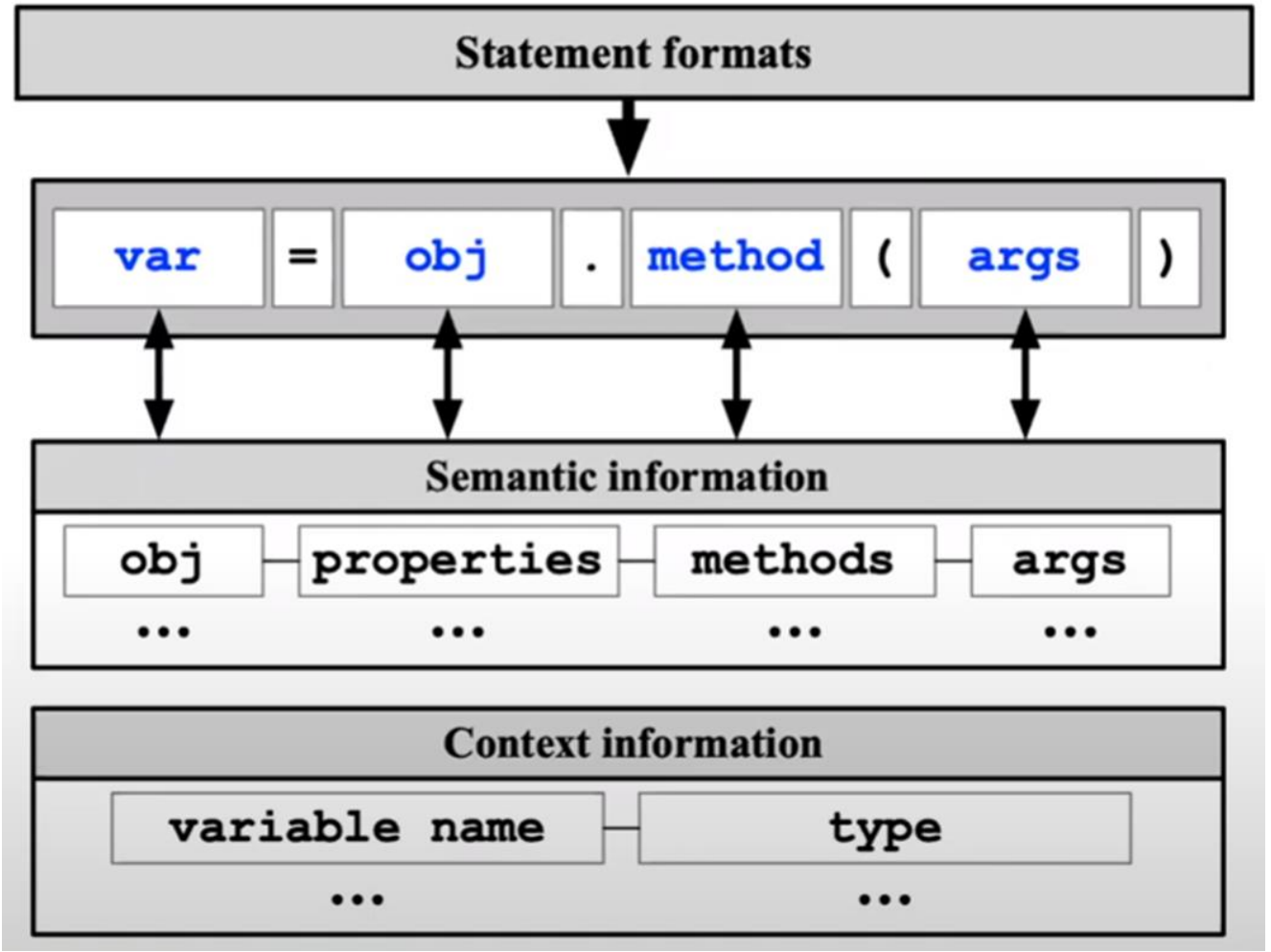
FAVOCADO-GENERATED JS FORMAT

Statement formats

```
 1 var obj = new obj(args)
 2 obj.prop = value
 3 var variable = obj.method_with_return(args)
 4 obj.method_without_return(args)
 5 for(var i=1; i++; i<n) { statements }
 6 array[index] = value
 7 obj.__proto__ = obj;
 8 obj.__defineSetter__(prop, func)
 9 obj.__defineGetter__(prop, func)
10 obj.prototype.method()
11 function(args) { statements }
```

TÉCNICO LISBOA

GENERATING JS STATEMENT FOR FUSSING



TÉCNICO LISBOA

EVALUATION

TÉCNICO LISBOA

FAVOCADO-EVALUATION

- Targeted JS Runtime Systems and Binding Objects

PDF

PDF

Mojo and DOM

DOM

- Counting distinct bugs
  - Used the most recent version of target systems, all bugs found by Favocado were previously unknown ones

TÉCNICO LISBOA

FAVOCADO-EVALUATION RESULT

Target System	Binding Object	# of VMs used	Fuzzing Duration	# of Bugs	# of Vulnerabilities
Adobe Acrobat Reader	PDF	8	2 weeks	45	24
Foxit Reader	PDF	8	3 days	3	3
Chromium	Mojo	8	1 week	2	1
Chromium	DOM	8	2 weeks	6	2
WebKit	DOM	8	4 days	5	3
			Total	**62**	**34**

- 2 cores and 4GB of memory for each VM (1 fuzzing process executes on each VM)
- 13 vulnerabilities have become CVE entries as of today

TÉCNICO LISBOA

COMPARISON

TÉCNICO LISBOA

CONCLUSION

- Proposed Favocado, a JavaScript binding code fuzzer, that can generate semantically correct test cases.
- Demonstrated the importance of semantically correct test cases and the effectiveness of Favocado.

TÉCNICO LISBOA

OBSERVATION

1.Fuzzing process can be more efficient.

2.Favocado does not rely on a specific feature of JavaScript.

3. They could not completely avoid manual effort to construct the semantic information of binding code that Favocado can process.

4.Memory related errors, such as memory leakage is not explained

TÉCNICO LISBOA

THANK YOU

joseph.thachil.george@tecnico.ulisboa.pt