FUZZING JAVASCRIPT ENGINES
FOR FUN & PROFIT

AREUM LEE@SSG
SINGI@THEORI
AREUM LEE

- Areum Lee
- Member @ SSG
- Undergrad student @ Sejong Univ
- Former intern @ Fortinet HQ
- Alumnus @ BoB
SINGI

- Jeonghoon Shin
- Member @ SSG
- Researcher @ Theori
- Mentor @ BoB
- Full time daddy
CONTENTS

1. Overview
   ▸ What we aimed to do
   ▸ How we did it

2. Our Fuzzer
   ▸ Characteristics & Environment
   ▸ Overall structure
   ▸ JFF
   ▸ Fuzzer
     - Bella
     - Benjamin

3. Conclusion
   ▸ Result
   ▸ Limitations
WHAT WE AIMED TO DO

1. Find vulnerabilities in browser javascript engines!
   - v8
   - javascript core
   - chakra core
   - spider monkey

2. Utilize fuzzer!
WHY TARGET BROWSERS?

- Everyone uses web browsers.
  - If a browser is vulnerable, a lot of people are prone to attacks.
- Web standards are continuously being updated. New features are added continuously.
  - More changes to code. So more chance of bugs?
- Web browser security is super important.
WHY JAVASCRIPT?

- Javascript is easier to exploit compared to DOM objects

```html
<html>
<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>
<body onunload="start();">
</body>
</html>
```

Comment 26 by e...@chromium.org, Mar 7 2017

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

- Documentation of ECMA Script is well maintained
- If a zero-day is found, it will work on similar js engine versions
WHY FUZZING?

- It saves time!
  - It creates many test cases in a short time
  - You can focus on other work while the computer runs the fuzzer.
  - If you can’t find bugs via source code auditing, you have somewhere to turn to. 😜
HOW WE DID IT

- Create a Javascript Fuzzing Factory
- Manage fuzzing nodes using Docker
- Make fuzzer create test cases based on existing 1day cases.
- Test case does not need to have any meaning to it. Just needs to create crashes!
CHARACTERISTICS & ENVIRONMENT

[ CHARACTERISTICS ]

▷ Fuzzing
  • In-memory fuzzing

▷ Management
  • use Docker!

▷ Creation of Test Case
  • Mutation based on existing 1days
  • Generation using dictionary for javascript syntax

[ ENVIRONMENT ]

▷ Fuzzing Server
  • Amazon EC2 service
  • 8 GB RAM
  • 4 CPU core
OVERALL STRUCTURE

1. JavaScript Fuzzing Factory
2. Management
3. Docker
4. Fuzzer
5. Bella
6. Benjamin
7. JS Engine
8. Crash Collector
9. Crash Collector

Diagram shows the flow from Bella and Benjamin fuzzing into the JS Engine, leading to crashes that are collected.
JFF - JAVASCRIPT FUZZING FACTORY

JAVA SCRIPT FUZZING FACTORY

MANAGEMENT PLATFORM

DOCKER

NODE 1
NODE 2
NODE 3
NODE 4
JFF – JAVASCRIPT FUZZING FACTORY

- Comprises of:

1. Docker
   - js engines and fuzzer run within the docker nodes

2. Web management platform
   - Use node js API to control the docker
JFF - JAVASCRIPT FUZZING FACTORY

- In-memory fuzzing

- make changes to javascript engine code

```javascript
if (isModule) {
    promise = loadAndEvaluateModule(globalObject->globalExec(), file
    scope.releaseAssertNoException();
} else {
    if (!fetchScriptFromLocalFileSystem(fileName, scriptBuffer))
        return false; // fail early so we can catch missing files
```
In-memory fuzzing

```csharp
static bool fetchScriptFromLocalFileSystem(const String& fileName, buffer)
{
    if (!fillBufferWithContentsOfFile(fileName, buffer))
        return false;
    convertShebangToJSCComment(buffer);
    return true;
}

static bool fillBufferWithContentsOfFile(FILE* file, Vector<char> & buffer)
{
    // We might have injected "use strict"; at the top.
    size_t initialSize = buffer.size();
    fseek(file, 0, SEEK_END);
    size_t bufferCapacity = ftell(file);
    fseek(file, 0, SEEK_SET);
    buffer.resize(bufferCapacity + initialSize);
    size_t readSize = fread(buffer.data() + initialSize, 1, buffer.size(),
                            return readSize == buffer.size() - initialSize;
```
JFF - JAVASCRIPT FUZZING FACTORY

- In-memory fuzzing

- Before

```cpp
} else {
    std::string source_code;
    if (!fetchScriptFromLocalFileSystem(fileName, scriptBuffer))
        return false; // fail early so we can catch missing files
}
```

- After

```cpp
} else {
    std::string source_code;
    //if (!fetchScriptFromLocalFileSystem(fileName, scriptBuffer))
    for(std::string line; std::getline(std::cin, line);) {
        source_code += line + "\n";
    }
    char *writable = new char[source_code.size() + 1];
    std::copy(source_code.begin(), source_code.end(), writable);
    writable[source_code.size()] = \0';
    scriptBuffer.append(writable, strlen(writable));
    //return false; // fail early so we can catch missing files
```
**JFF – JAVASCRIPT FUZZING FACTORY**

- In-memory fuzzing
  - Before
    ```
    singiui-MacBook-Air:~ singi$ iostat
disk0  cpu  load average
    KB/t  tps  MB/s  us  sy  id  1m  5m  15m
    46.58 11  0.48  16  3  81  3.39  3.25  2.91
    ```
  - After
    ```
    singiui-MacBook-Air:Safari-604.5.6 singi$ iostat
disk0  cpu  load average
    KB/t  tps  MB/s  us  sy  id  1m  5m  15m
    45.97 11  0.48  16  4  80  2.00  2.21  2.44
    ```
THE FUZZERS

BELLA

LEGO

BENJAMIN

NULL
BELLA

- Mutation based
  - Apply mutations on existing 1days by finding patterns
- Why?
  - There are too many cases.
  - Hard to find pattern by hand.
- How?
  - Create template from existing 1days.
  - Make minimal changes to create random JS file.
LEGO

- Our approach to create JS syntax
- Name given to the template created from 1 days.
- Parse 1-day PoCs for making LEGO file
- Parse LEGO file to make a new JS file.
- Excluded whatever was not important
BENJAMIN

- Generation based
  - make by using input grammar

- How?
  - Create input grammar by using library
Problem!

Test cases have fixed form

Have to be randomized

Solution

Make API!
**API**

```python
def setProp(self, retStr=False):
    r = ""
    obj = self.objectList
    if obj['type'] == objectType.Array:
        prop = choice(JSArrayObject().properties)
        propVar = Util.getVar(self.objectList, prop['type'])
        r += "%s.%s = %s" % (obj['name'], prop['name'], propVar)
    if retStr:
        return e(r)
    self.testcase += e(r)

def getProp(self, retStr=False):
    r = ""
    obj = self.objectList
    if obj['type'] == objectType.Array:
        prop = choice(JSArrayObject().properties)
        propVar = Util.getVar(self.objectList, prop['type'])
        r += "%s.%s" % (obj['name'], prop['name'])
    if retStr:
        return e(r)
    self.testcase += e(r)
```
if __name__ == '__main__':
    # for API test
    fuzz = tejava("jsc1.js")
    for i in range(5):
        fuzz.createObject()

    fuzz.JSfor(countVar='i', funcs=[fuzz.JSdelete, fuzz.createObject, fuzz.setProp, fuzz.getProp, callMethodGlobal, fuzz.callFunction])
    fuzz.setVar()
    fuzz.JSgetterOrsetter()
    for i in range(5):
        fuzz.getVar()

    fuzz._print()
CRASH COLLECTOR

- Use regex

\s\*[0-9A-F]{8}\s\|.\*\.[\(eax|ebx|ecx|edx|esi|edi|ebp|esp|eip)\.*\]

0x008: x86_

eax=00000000 ebx=059e47a8 ecx=059e47a8 edx=02ad7170 esi=039cf814 edi=059e47a8
eip=50b21d22 esp=039cf4e0 ebp=039cf50c iopl=0 nv up ei pl zr na pe nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00010246
MSHTML!Layout::FlowBoxBuilder::SContentReader::TransitionBuilderIntoBuildingLine+0x2c0:
50b21d22 8b4018 mov eax, dword ptr [eax+18h] ds:002b:00000018=????????
OVERALL STRUCTURE (AGAIN)
IT'S LIKE AN ICE CREAM SUNDAE!
RESULT

- Crash produced by using Benjamin.
- Target: Safari javascript core

- Occurred in web assembly due to overflow.
DEMO
WE’RE STILL FAR FROM PERFECT

- We need good code coverage.
- JFF is small-scale compared to those being used in big companies.
- Limitations to creating different types of JS templates.
  - The sequence of API usage may be limited
- We have few minor bugs.
FUTURE PLANS

- Keep track of ECMA Script updates and add to fuzzer
  - ECMA script updates will also be applied to javascript engine
- Enhance JFF to support other vectors
Q&A