KCon 洞见未来 2017

KCon

Webkit Vulnerability form 0 to 1

Lockmanxxx

who

- Work for 兴华永恒
- 软件漏洞利用研究
- 软件漏洞成因分析
- 偶尔漏洞挖掘

@weibo:Lockmanxxx
@twitter:lockmanxxx

高效的挖掘？！？

From 0 to 1

PART 01 浏览器漏洞挖掘难点

PART 02 Why WebKit ?

PART 03 挖掘思路

PART 04 挑战

PART 05 漏洞挖掘平台

PART 06 展示

PART 07 To Do Soon

目录
CONTENTS

01 浏览器漏洞挖掘难点

1.重要攻击向量，防之又防

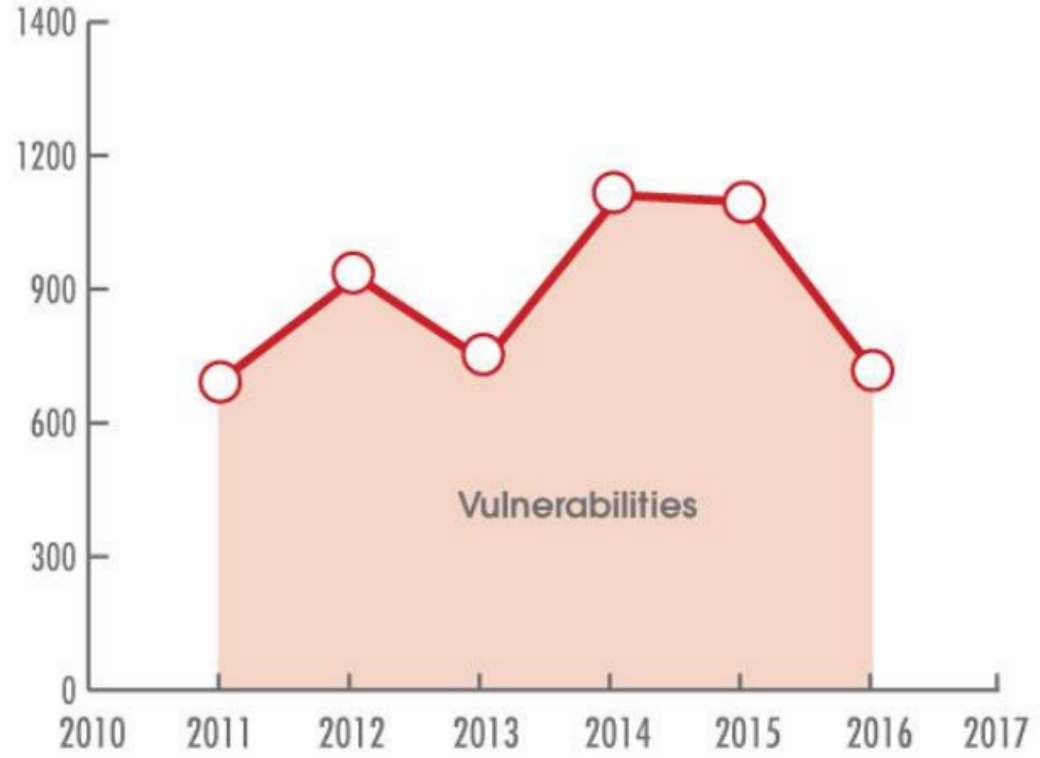
2.软件设计复杂，代码关系复杂

3.各个版本各有特点

4.从设计角度阻碍漏洞挖掘

Figure 22

VULNERABILITIES IN THE TOP 5 MOST POPULAR BROWSERS



Source: Vulnerability Review 2017

https://www.flexerasoftware.com/enterprise/resources/research/vulnerability-review/

Browsers

- Webkit
  - Blink(chrome)、Apple Safari
- Gecko
  - Fireforx
- Trident
  - Internet Explorer
- JavaScript Engine也开始出现分化。

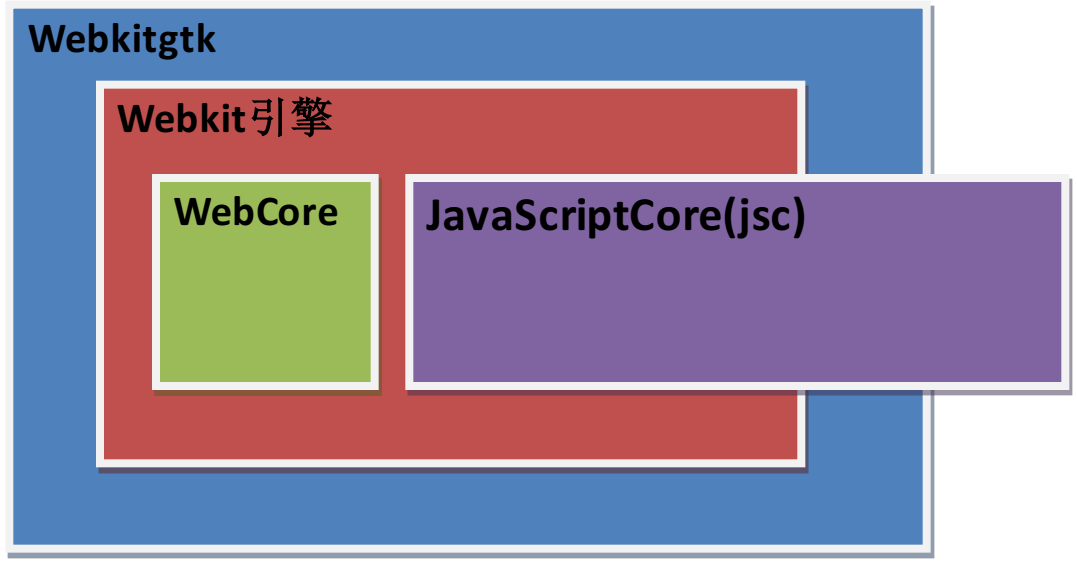
02 Why Webkit?

特点

- 开源
- “Chrome”,safari, Ucbrowser, QQ Browser……
- 多平台支持，Android+windows+linux
- WebKitGTK

WebKitGtk

- Miniborwser
- WebKitWebProcess
- ~~WebKitNetworkProcess~~
- JSC



03 挖掘思路

首先

$+ (E=mc^2) =>$

其次

详细

- 代码分析（不深入）
  - “铀”
- 根据分析搜集的数据，定制fuzz输入
  - “E=mc2”
- 随机规则
- 挖掘平台
- 根据代码覆盖率，不断完善。

04

挑战☺

挖掘对象

网页模型

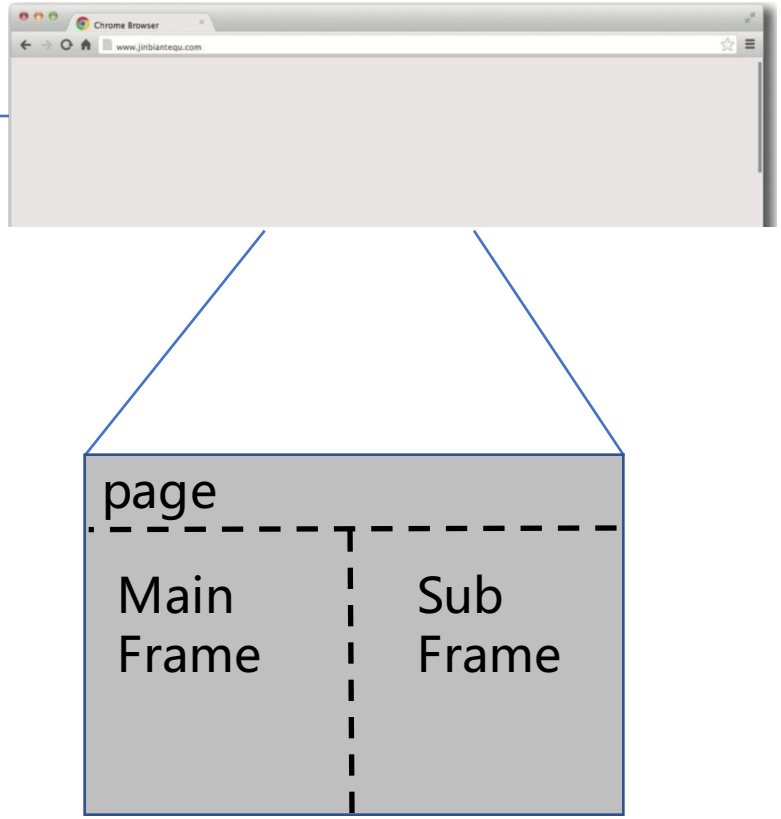

CSS
SVG
MathML
NPAPI
Geo
Fetch
JPEG
GIF
CSP
PNG
THIS SPECIFICATION
HTTP
TLS
MQ
DOM
Unicode
Web
IDL
MIME
URL
XML
JavaScript
Encodings

WebKit parser

- Html
  - -解释型
  - -DOM树
- JavaScript
  - -脚本语言
  - -词法-语法-语义-字节码-编译-优化-汇编

HTML parser

Code Flow
WebKitWebView
WebCore::Page
WebCore::FrameLoader
WebCore::MainFrame
WebCore::DocumentLoader
WebCore::HTMLDocument
WebCore::DOMWindow
WebCore::HTMLDocumentParser
WebCore::HTMLScriptRunner



HTML parser

- HTML→HTMLToken (标签提取)
  - https://html.spec.whatwg.org/
  - 根据标签的开闭合字符逐个提取Token
    HTMLDocumentParser::pumpTokenizerLoop
    HTMLTokenizer::processToken
  - 类似字符串整理，表现形式为vector对象链表。

HTML parser

- processToken

```
1   BEGIN_STATE(DataState)
2           if (character == '&')
3   ADVANCE_PAST_NON_NEWLINE_TO(CharacterReferenceInDataState);
4           if (character == '<') {
5   ......
6   BEGIN_STATE(TagOpenState){          //继续跳转到其它Case
7   ......
8   BEGIN_STATE(EndTagOpenState){
9   ......
10          if (character == '>') {
11  ......
12  BEGIN_STATE(TagNameState){
13  ......
```

HTML parser

- HTMLToken→DOMTree
  - 根据前面提取信息进行逐个解析

  HTMLTreeBuilder::processToken

```
1   case HTMLToken::StartTag:
2           m_shouldSkipLeadingNewline = false;
3           processStartTag(WTFMove(token));
4           break;
```

- HTMLToken→DOM Node

  WebCore::HTMLConstructionSite

评估

- HTML parser
  - 过程单一
  - 解释性操作
  - 对HTML来说重要的仅仅就是标签的开、闭合。

HTML Parser ☹

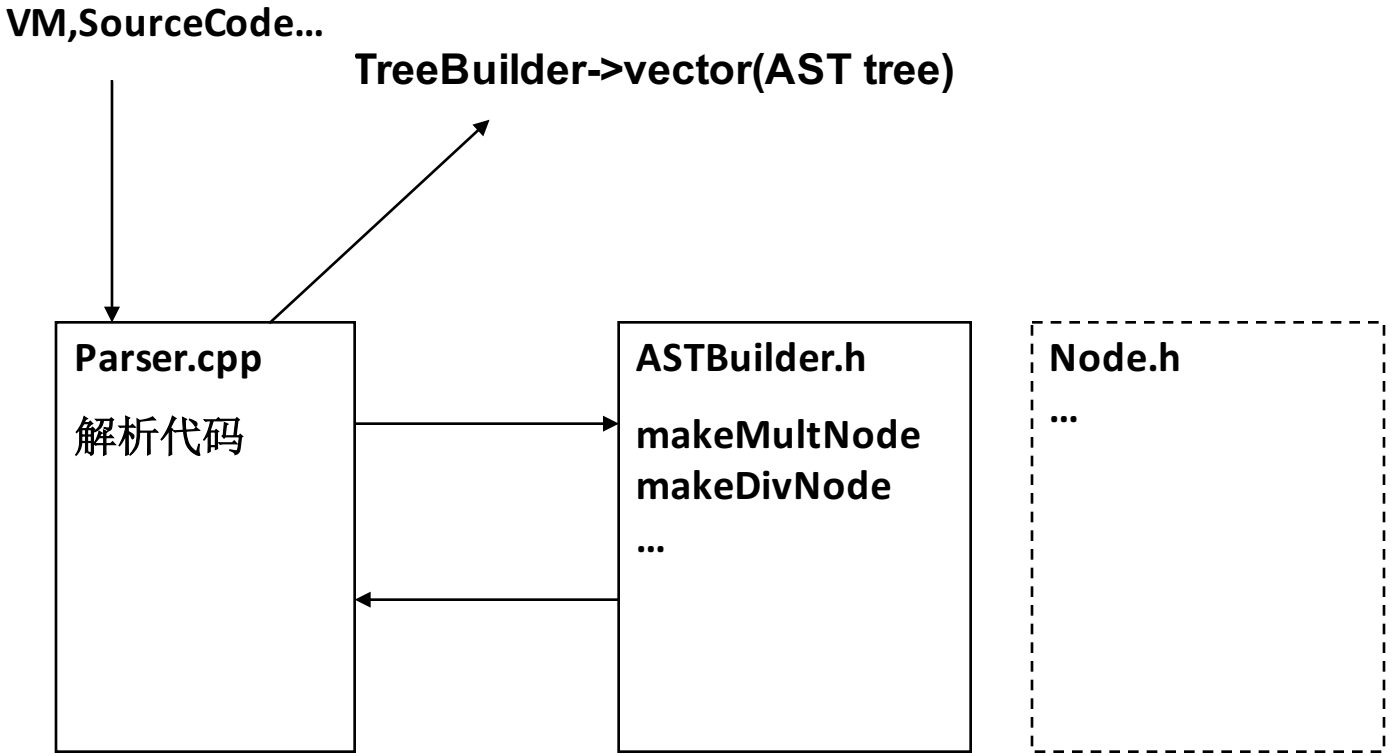
JavaScript Parser

- JavaScript
  - ./source/JavaScriptCore/parser
  - JavaScript parser入口

```
1   bool HTMLDocumentParser::pumpTokenizerLoop(SynchronousMode mode,
2                                              bool parsingFragment,
3                                              PumpSession& session)
4   {
5       do {
6           if (UNLIKELY(isWaitingForScripts())) {
7               if (mode == AllowYield && m_parserScheduler->shouldYieldBeforeExecutingScript(session))
8                   return true;
9               runScriptsForPausedTreeBuilder();
10
11              if (isWaitingForScripts() || isStopped())
12                  return false;
13          }
```

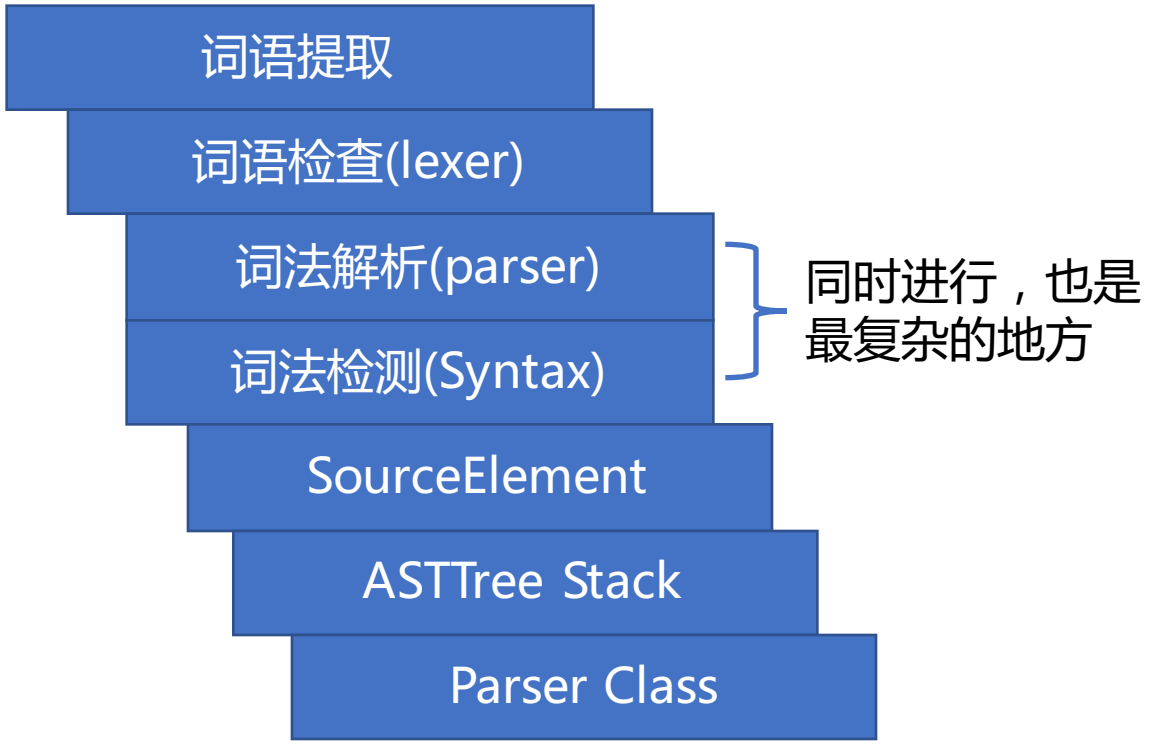
JavaScript Parser

- 工作流



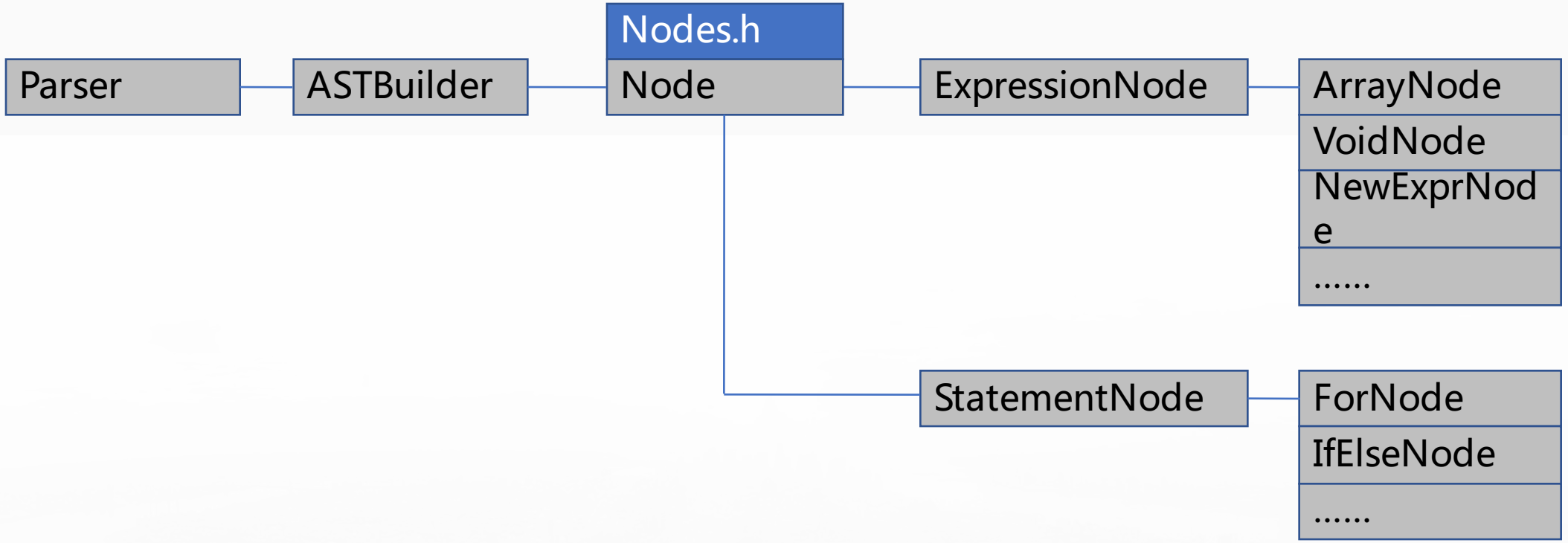
JavaScript Parser

- 解析流程



JavaScript Parser

- 对象地图



JavaScript Parser

- 数据来源

```
1   template <class ParsedNode>
2   std::unique_ptr<ParsedNode> parse(
3       VM* vm, const SourceCode& source,
4       ......)
```

代码source.provider()返回Class SourceProvider对象。

```
multi-thre Thread 0x7f91d10f0a In: JSC::parse<JSC::P
(gdb) p source.provider()
$1 = (JSC::SourceProvider *) 0x7f91b4ff0c60
(gdb)
```

而source.provider()->source()继续返回一个StringView对象。

```
(gdb) p source.provider()->source()
$2 = {m_characters = 0x7f91c730b228, m_length = 112, m_is8Bit = true, m_underlyingString = 0xc34190}
(gdb)
```

JavaScript Parser

- 数据来源

StringView是webkit封装的一个字符串类，其构造函数：

```
inline StringView::StringView(const StringImpl& string)
{
    setUnderlyingString(&string);
    if (string.is8Bit())
        initialize(string.characters8(), string.length());
    else
        initialize(string.characters16(), string.length());
}
```

此时能看到<script>标签中所输入的代码：

```
KindCount = 4, static s_hashFlagStringKindIsAtomic = 16,
 s_hashFlagDidReportCost = 4, static s_hashMaskBufferOwnership = 3,
8a u"[redacted]"},          m_hashAndFlags = 992094104},
Count = 6, m_length = 17, {m_data8 = 0x7f91b4e58f3c "\n//\nvar testb=5;\n",
```

JavaScript Parser

- Js Parser解析详情

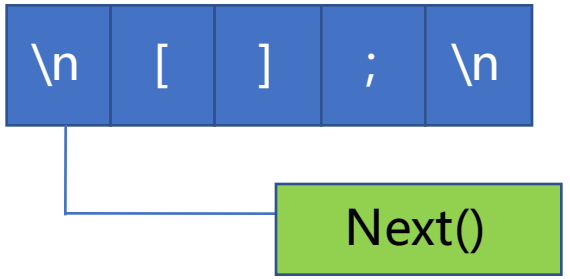
前面的SourceCode对象首先和Parser::m_lexer建立联系，通过：

```
m_lexer->setCode(source, &m_parserArena);
```

如果原始网页js代码为下面的例子：

```
<script>
[];
</script>
```

按JS Parser解析逻辑会逐字拆分：



JavaScript Parser

- Js Parser解析详情

  Next函数让JS Parser知道了下一步处理的位置和代码等信息：

```
ALWAYS_INLINE void next(unsigned lexerFlags = 0)
    {
        int lastLine = m_token.m_location.line;
        int lastTokenEnd = m_token.m_location.endOffset;
        int lastTokenLineStart = m_token.m_location.lineStartOffset;
        m_lastTokenEndPosition = JSTextPosition(lastLine, lastTokenEnd, lastTokenLineStart);
        m_lexer->setLastLineNumber(lastLine);
        m_token.m_type = m_lexer->lex(&m_token, lexerFlags, strictMode());
    }
```

JavaScript Parser

- Js Parser解析详情

每个字符的属性(type)定义：

```
// 256 Latin-1 codes
static const unsigned short typesOfLatin1Characters[256] = {
/*   0 - Null                */ CharacterInvalid,
/*   1 - Start of Heading    */ CharacterInvalid,
/*   2 - Start of Text       */ CharacterInvalid,
/*   3 - End of Text         */ CharacterInvalid,
/*   4 - End of Transm.      */ CharacterInvalid,
/*   5 - Enquiry             */ CharacterInvalid,
/*   6 - Acknowledgment      */ CharacterInvalid,
/*   7 - Bell                */ CharacterInvalid,
/*   8 - Back Space          */ CharacterInvalid,
/*   9 - Horizontal Tab      */ CharacterWhiteSpace,
/*  10 - Line Feed           */ CharacterLineTerminator,
/*  11 - Vertical Tab        */ CharacterWhiteSpace,
/*  12 - Form Feed           */ CharacterWhiteSpace,
/*  13 - Carriage Return     */ CharacterLineTerminator,
/*  14 - Shift Out           */ CharacterInvalid,
/*  15 - Shift In            */ CharacterInvalid,
/*  16 - Data Line Escape    */ CharacterInvalid,
/*  17 - Device Control 1    */ CharacterInvalid,
/*  18 - Device Control 2    */ CharacterInvalid,
/*  19 - Device Control 3    */ CharacterInvalid,
/*  20 - Device Control 4    */ CharacterInvalid,
/*  21 - Negative Ack.       */ CharacterInvalid,
```

JavaScript Parser

- Js Parser解析详情

  lex函数最终把每个词语提取成JSTokenType：

```
1   ......
2   case   CharacterOr:              //it means '|'
3            shift();
4            if (m_current == '=') {
5                shift();
6                token = OREQUAL;
7                break;
8            }
9            ......
10           token = BITOR;
11           break;
12  ......
13  return token;
```

JavaScript Parser

- Js Parser解析详情

最终的JSTokenType就是parser逻辑主体所处理的内容：

```
enum JSTokenType {
    NULLTOKEN = KeywordTokenFlag,
    TRUETOKEN,
    FALSETOKEN,
    BREAK,
    CASE,
    DEFAULT,
    FOR,
    NEW,
    VAR,
    CONSTTOKEN,
    CONTINUE,
    FUNCTION,
    RETURN,
    IF,
```

对于JS代码“[];”来讲，在预处理代码为JSTokenType后，其最主体的处理函数parseArrayLiteral ：

JavaScript Parser

- Js Parser解析详情

```
template <typename LexerType>
template <class TreeBuilder> TreeExpression Parser<LexerType>::parseArrayLiteral(TreeBuilder& context)
{
    consumeOrFailWithFlags(OPENBRACKET, TreeBuilder::DontBuildStrings, "Expected an opening '[' at the beginning of an array literal");

    int oldNonLHSCount = m_parserState.nonLHSCount;

    int elisions = 0;
    while (match(COMMA)) {
        next(TreeBuilder::DontBuildStrings);
        elisions++;
    }
    if (match(CLOSEBRACKET)) {
        JSTokenLocation location(tokenLocation());
        next(TreeBuilder::DontBuildStrings);
        return context.createArray(location, elisions);
    }

    TreeExpression elem;
    if (UNLIKELY(match(DOTDOTDOT))) {
        auto spreadLocation = m_token.m_location;
        auto start = m_token.m_startPosition;
        auto divot = m_token.m_endPosition;
        next();
        auto spreadExpr = parseAssignmentExpressionOrPropagateErrorClass(context);
        failIfFalse(spreadExpr, "Cannot parse subject of a spread operation");
        elem = context.createSpreadExpression(spreadLocation, spreadExpr, start, divot, m_lastTokenEndPosition);
    } else
        elem = parseAssignmentExpressionOrPropagateErrorClass(context);
    failIfFalse(elem, "Cannot parse array literal element");
    typename TreeBuilder::ElementList elementList = context.createElementList(elisions, elem);
    typename TreeBuilder::ElementList tail = elementList;
    elisions = 0;
    while (match(COMMA)) {
        next(TreeBuilder::DontBuildStrings);
        elisions = 0;

        while (match(COMMA)) {
            next();
            elisions++;
        }

        if (match(CLOSEBRACKET)) {
```

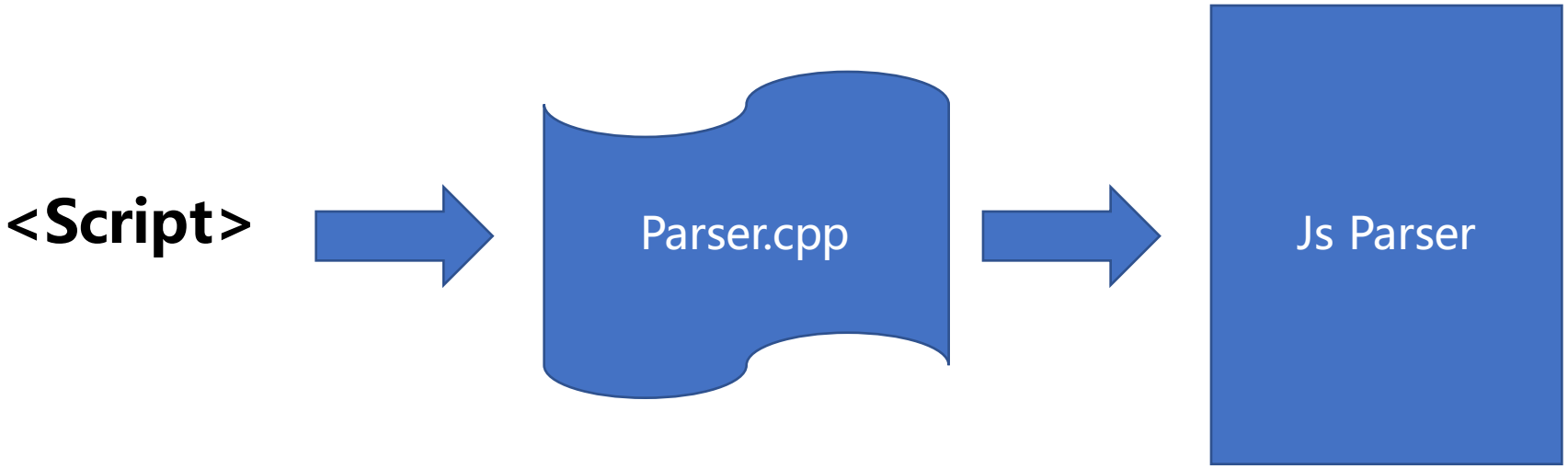
评估

- Javascript Parser

  - ✓ 逻辑较复杂。
  - ✓ 对象关系复杂。
  - ✓ 可编程语言JavaScript，可变因素很多。
  - ✓ 初步分析，已经能直观建立用户输入与代码逻辑的联系。

JavaScript Parser ☺

评估



eg: parseArrayLiteral函数逻辑的全代码覆盖：

```
while (match(COMMA)) {
……
if (match(CLOSEBRACKET)) {
……
if (UNLIKELY(match(DOTDOTDOT))) {
…….
while (match(COMMA)) {
```

评估

数据

[]
[...]
[,1,,1,,]
[,,,,]
[1,1,1]

05 漏洞挖掘平台

数据输入

+ $(E=mc^2)$

File Edit Format View Help

```
?
:
=
+=
-=
*=
%=
<<=
>>=
>>>=
&=
|=
^=
=>
`
"
'
}
;
/
/=
break
do
in
typeof
case
else
instanceof
var
catch
new
void
class
```

其他搜集的数据

数据输入

- 以JS parser模块为例
  - {0, void :*++=++%a14074,17957, void swv>><<=l%!uphs-=h213, this t, const r,3334}
  - yield !t+?=&&

V={=>

  - https://bugs.webkit.org/show_bugs?id=147538

- 其他类型数据

```
<script>
function testMathyFunction(f, inputs)
{
  var results = [];
  if (f) {
    for (var j = 0; j < inputs.length; ++j) {
      for (var k = 0; k < inputs.length; ++k) {
        try {          results.push(f(inputs[j], inputs[k]));
        } catch(e) {
          results.push(e);
        }
      }
    }
  }
}
var mathy2=(function(x, y) { "use strict";
return ((this >>> 0) ? (this | 0) : (Math.ceil((+ Math.sqrt(((Math.fround((mathy2((this >>> 0), this) | 0)) || Math.fround(this)) >>> 0)))) >>> 0)); })
testMathyFunction(mathy2, [-(2**53),42,-(2**53),1,-0,0/0]);
</script>
```

挖掘平台

- AFL is Everything! But ……
- 样本的恢复，提取（代码性质样本）
- Js代码的随机规则
- 编译感知
- 覆盖率工具，反馈

挖掘平台

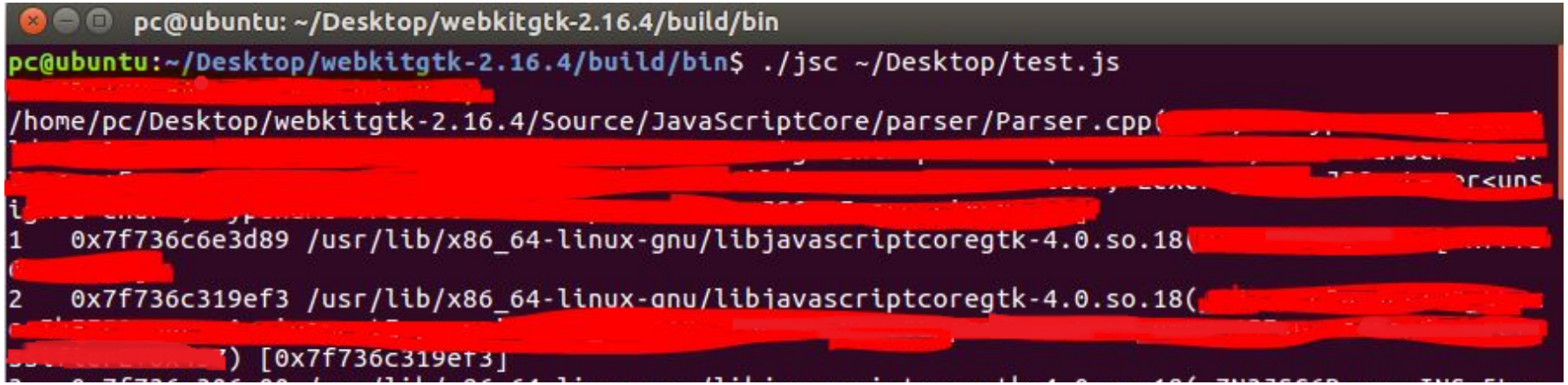
+ $(E=mc^2)$

```
 process timing                                 overall results
        run time : 0 days, 23 hrs, 2 min, 7 sec  cycles done : 9
   last new path : 0 days, 4 hrs, 1 min, 33 sec  total paths : 481
 last uniq crash : 0 days, 3 hrs, 21 min, 29 sec uniq crashes : 1
  last uniq hang : 0 days, 3 hrs, 4 min, 24 sec   uniq hangs : 48
 cycle progress                                 map coverage
  now processing : 460* (95.63%)                  map density : 1.05% / 1.48%
 paths timed out : 0 (0.00%)                    count coverage : 2.40 bits/tuple
 stage progress                                 findings in depth
  now trying : interest 32/8                    favored paths : 53 (11.02%)
 stage execs : 398/4735 (8.41%)                  new edges on : 74 (15.38%)
 total execs : 3.53M                            total crashes : 1 (1 unique)
  exec speed : 2.52/sec (zzzz...)                total tmouts : 35.1k (60 unique)
 fuzzing strategy yields                        path geometry
   bit flips : 180/112k, 57/111k, 42/111k         levels : 18
  byte flips : 5/14.0k, 0/13.5k, 3/12.8k          pending : 114
 arithmetics : 69/774k, 8/137k, 1/28.0k          pend fav : 0
  known ints : 5/75.7k, 17/355k, 5/553k         own finds : 475
  dictionary : 0/0, 0/0, 19/532k                 imported : 0
       havoc : 65/683k, 0/0                     stability : 99.38%
        trim : 70.00%/3922, 0.77%
                                                          [cpu:302%]
writelen 0x000056b8
leftlen 0x00001000e
loopnum 1x00004000e
writelen 0x000056b8
leftlen 0x00002000e
```

06 展示

崩溃

1


pc@ubuntu: ~/Desktop/webkitgtk-2.16.4/build/bin
pc@ubuntu:~/Desktop/webkitgtk-2.16.4/build/bin$ ./jsc ~/Desktop/test.js
/home/pc/Desktop/webkitgtk-2.16.4/Source/JavaScriptCore/parser/Parser.cpp
1 0x7f736c6e3d89 /usr/lib/x86_64-linux-gnu/libjavascriptcoregtk-4.0.so.18
2 0x7f736c319ef3 /usr/lib/x86_64-linux-gnu/libjavascriptcoregtk-4.0.so.18(

2


Starting program: /home/pc/Desktop/webkitgtk-2.16.4/build/bin/jsc ~/Desktop/test.js
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[New Thread 0x7fffefeeb700 (LWP 3746)]
[New Thread 0x7fffedbff700 (LWP 3747)]
Thread 1 "jsc" received signal SIGTRAP, Trace/breakpoint trap.
in ?? ()
(gdb) i s 10
#0 in ?? ()
#1 in ?? ()
#2 in ?? ()
(gdb)

结果

2 weeks

数据+测试+修改

10+Crashes

Some are interesting

3 moudles

Parser+Runtime+API

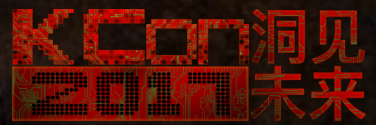
07 To Do Soon

计划

更多的自动化

无源码项目？

Question?!



Thank you!

KCon 洞见未来 2017