

PHP是如何解析JSON的

[virink](#) / 2019-04-11 08:40:00 / 浏览数 3766 [安全技术](#) [WEB安全](#) [顶\(0\)](#) [踩\(0\)](#)

Info

```
json_decode ( string $json [, bool $assoc = false [, int $depth = 512 [, int $options = 0 ]]] ) : mixed
```

(PHP 5 >= 5.2.0, PHP 7, PECL json >= 1.2.0)

json_decode — 对 JSON 格式的字符串进行解码

参数	说明
json	待解码的 json string 格式的字符串。(RFC 7159)
assoc	当该参数为 TRUE 时, 将返回 array 而非 object.
depth	指定递归深度。
options	JSON解码的掩码选项。 现在有两个支持的选项。 JSON_BIGINT_AS_STRING, 大整数转为字符串 (默认float类型)。 JSON_OBJECT_AS_ARRAY, 与将assoc设置为 TRUE 有相同的效果。

变化

- 自 PHP 5.2.0 起, JSON 扩展默认内置并编译进了 PHP。
[PHP 5 JSON checker - Douglas Crockford](#)
 - a Pushdown Automaton that very quickly determines.
- PHP 7 中是改进的全新解析器, 专门为 PHP 订制, 软件许可证为 PHP license.
 - re2c 0.16
 - Bison 3.0.4

PHP 版本说明

- PHP 5 [5.6.40](#)
- PHP 7 [7.3.4](#)

PHP 5 : Call Stack of json_decode (zif_json_decode)

毕竟5已经停止更新了, 就简单提及一下

```
zif_json_decode ext/json/json.c:831-857
  php_json_decode_ex ext/json/json.c:680-796
    • json_utf8_to_utf16 utf8 转 utf16
    • new_JSON_parser 初始化
    • parse_JSON_ex 解析 ext/json/JSON_parser.c:439-750
```

JSON_parser.c 就是 JSON_checker 的

PHP 7 : Call Stack of json_decode (zif_json_decode)

zif_json_decode

ext/json/json.c:312-362

默认嵌套深度 : depth = PHP_JSON_PARSER_DEFAULT_DEPTH (= 512 [php_json.h])
最大嵌套深度 : depth > INT_MAX (= 2147483647 [php.h])

接受参数并调用php_json_decode_ex

php_json_decode_ex

ext/json/json.c:246-264

- 初始化 PHP 的json解析器 php_json_parser_init
- 解析 json 字符串 php_json_yyparse
- 如果解析错误, 抛出异常及错误信息
- 返回 PHP Object

php_json_yyparse (yyparse)

ext/json/json_parser.tab.c:115

```
#define yyparse php_json_yyparse
```

ext/json/json_parser.tab.c:1194-843

```
int yyparse/php_json_parser *parser)
```

```
// L:1359
```

```
if (yychar == YYEMPTY)
```

```
{
```

```
    YYDPRINTF((stderr, "Reading a token: "));
```

```
    yychar = yylex(&yyval, parser);
```

```
}
```

yylex (php_json_yylex)

ext/json/json_parser.tab.c:116

```
#define yylex php_json_yylex
```

ext/json/json_parser.tab.c:1899-1904

```
static int php_json_yylex(union YYSTYPE *value, php_json_parser *parser)
```

```
{
```

```
    int token = php_json_scan(&parser->scanner);
```

```
    value->value = parser->scanner.value;
```

```
    return token;
```

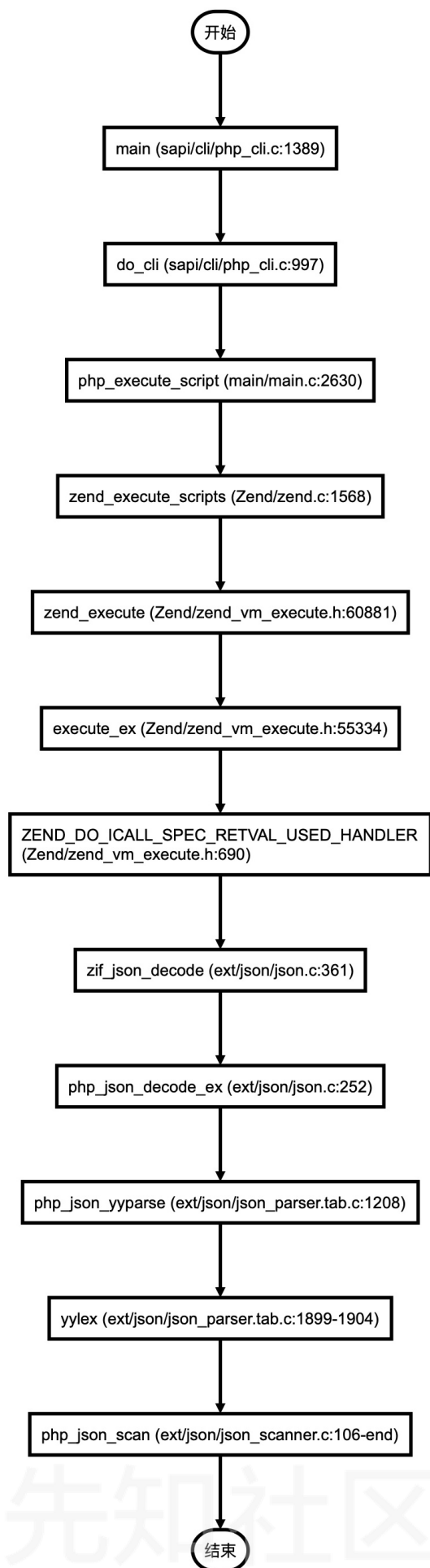
```
}
```

php_json_scan

ext/json/json_scanner.c:106-end

匹配并解析字符串 (包括\uXXXX,\r等)

Flowchart



Source Code json.php

```
<?php
print_r("==== Unicode \\u003e ===== ");
$str = '{"vk":"vir\\u003eink"}';
$obj = json_decode($str);
print_r($str . "\\r\\n");
print_r($obj);
/* Output
==== Unicode \\u003e =====
{"vk":"vir\\u003eink"}
stdClass Object
(
    [vk] => vir>ink
)
*/
```

[完整文件 - PHP是如何解析JSON的 - 测试文件 - Gist](#)

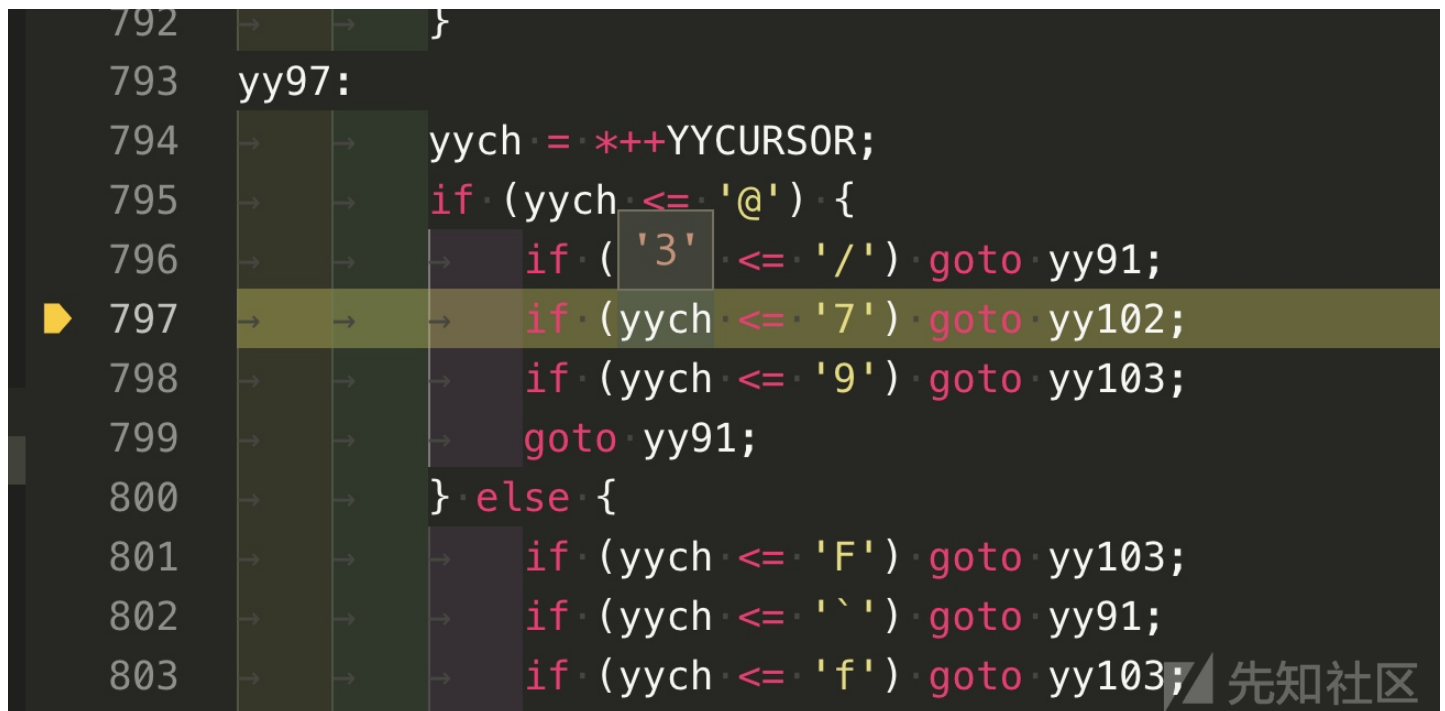
Call Stack

进入解析

1. php_json_scan (ext/json/json_scanner.c)

JSON字符 匹配 value "\\u003e"

1. 省略 vir...
2. if (yych <= '\\') goto yy77; (L:553) // \
3. if (yych <= 'u') goto yy90; (L:625) // u
4. if (yych <= '0') goto yy94; (L:706) // 0
5. if (yych <= '0') goto yy97; (L:747) // 0
6. if (yych <= '7') goto yy102; (L:797) // 3
7. if (yych <= 'f') goto yy107; (L:863) // e
8. s->str_esc += 5; (L:917) // \\u003e -> >
9. size_t len = s->cursor - s->str_start - s->str_esc - 1 + s->utf8_invalid_count; (L:583) // ■■■■
10. 省略ink



```
792     }
793 yy97:
794     yych = *++YYCURSOR;
795     if (yych <= '@') {
796         if ('3' <= '/') goto yy91;
797         if (yych <= '7') goto yy102;
798         if (yych <= '9') goto yy103;
799         goto yy91;
800     } else {
801         if (yych <= 'F') goto yy103;
802         if (yych <= '`') goto yy91;
803         if (yych <= 'f') goto yy103;
```

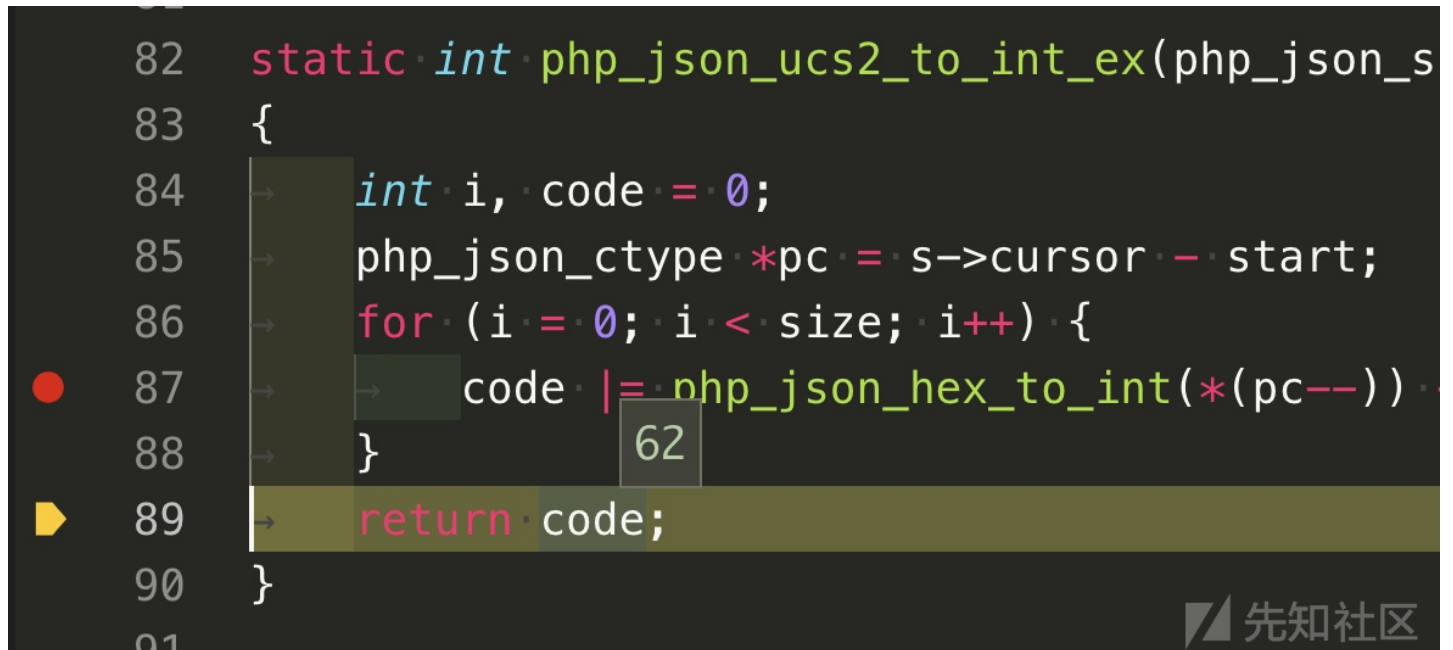
Unicode匹配并解析 (\\u003e)

1. if (yych == '\\') goto yy175; (L:1369) // \
2. if (yych == 'u') goto yy177; (L:1381) // u
3. if (yych <= '0') goto yy179; (L:1420) // 0
4. if (yych <= '0') goto yy182; (L:1443) // 0

```
5. if (yych <= '7') goto yy186; (L:1485) // 3
6. if (yych <= 'f') goto yy190; (L:1539) // e
```

转换 Unicode 为 Char

```
1. int utf16 = php_json_ucs2_to_int(s, 2); (L:1581->92)
2. return php_json_ucs2_to_int_ex(s, size, 1); (L:94) // return 62
```



```
82 static int php_json_ucs2_to_int_ex(php_json_s
83 {
84     int i, code = 0;
85     php_json_ctype *pc = s->cursor - start;
86     for (i = 0; i < size; i++) {
87         code |= php_json_hex_to_int(*(pc--))
88     }
89     return code;
90 }
91
```

从静态到动态

静态分析

代码阅读工具

- Sublime Text 3
- Visual Studio Code
 - ext: Lex Flex Yacc Bison
- Understand

分析

- 目标: JSON 扩展
- 目录: ext/json/
- 文件: ext/json/json.c
- 函数: static PHP_FUNCTION(json_decode)

函数名称都比较容易看懂, 编辑器大多数都有转到定义功能

- php_json_decode_ex
 - php_json_parser_init
 - php_json_yparse
 - ...etc

上文(PHP 7: Call Stack of json_decode)列的也比较详细了

一个字就是看

ext/json/README也是要先看看的

The parser is implemented using re2c and Bison. The used versions of both tools for generating files in the repository are following:

```
re2c 0.16
Bison 3.0.4
```

当然, 后缀为*.y和*.re的文件也说明了这个解析涉及扫描程序(scanner)和语法分析(parser)。

PS. 编译原理。。。反正我不懂

动态分析

通过静态分析，我们可以得到一些关键函数

- php_json_decode_ex
- php_json_yyparse
- yylex
- php_json_scan

给php_json_scan下断点，单步进入一把竣，调试过程中随时加断点

- php_json_scan
- std: (ext/json/json_parser.tab.c:115)
- yyc_JS: (ext/json/json_parser.tab.c:167)
- yyc_STR_P1: (ext/json/json_parser.tab.c:547)
- php_json_ucs2_to_int_ex
- php_json_hex_to_int

PS. 单步跳过有点坑，尽可能用单步调试和单步跳出

点击收藏 | 0 关注 | 1

[上一篇：Discuz!x3.4后台文件任意...](#) [下一篇：Discuz!x3.4后台文件任意...](#)

1. 1 条回复



[wonderkun](#) 2019-04-16 14:16:23

这玩意没有漏洞，还有有漏洞被你按下不表？

0 回复Ta

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)