

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



得 flag: hctf{1t_iz_4_4mall_tr1ck}

Web.25:神奇的数字

首先分析最后一步, `is_palindrome_number($req["number"])`要否, 即字符串不是回文数, 然后要满足 `intval($req["number"]) == intval(strrev($req["number"]))`, 这两处很可疑, 都是判断回文数, 为什么要判断两遍, 而且结果还要不一样? 我查了一下 `intval`, 发现 `intval` 会先用 `strtol`, 而 `strtol` 会跳过前面的空格字符。这说明原 `number` 中肯定有空白字符没有删, 导致两种判断方式出现差别。

但是前面明明用了 `trim` 了啊?

```
foreach([$_GET, $_POST] as $global_var) {  
    foreach($global_var as $key => $value) {  
        $value = trim($value);  
        is_string($value) && is_numeric($value) && $req[$key] = addslashes($value);  
    }  
}  
  
$n1 = intval($req["number"]);
```

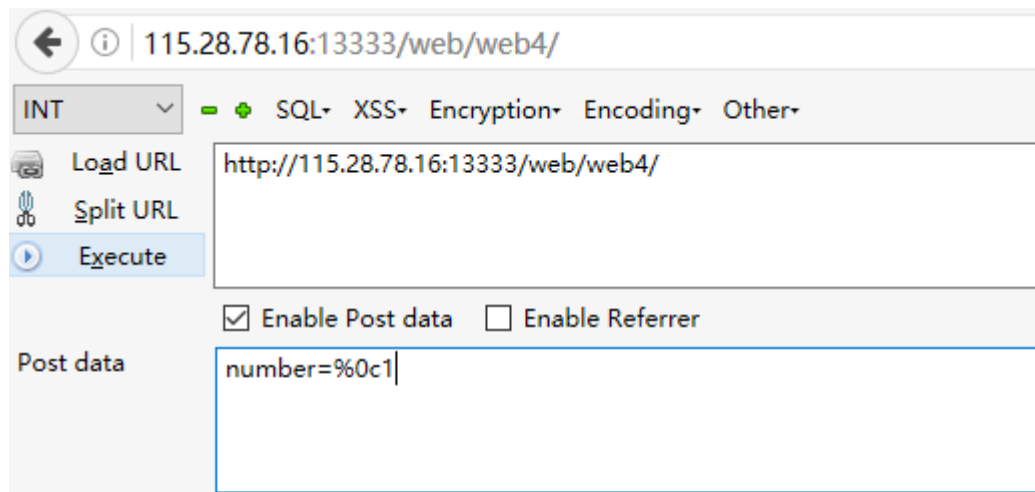
我又查了一下 `trim` 函数, 果然有漏洞:

语法

```
trim(string, charlist)
```

参数	描述
<i>string</i>	必需。规定要检查的字符串。
<i>charlist</i>	可选。规定从字符串中删除哪些字符。如果被省略, 则移除以下所有字符: <ul style="list-style-type: none">"\0" - NULL"\t" - 制表符"\n" - 换行"\x0B" - 垂直制表符"\r" - 回车" " - 空格

`Trim` 没有删掉字符串中的 `/f` 空白字符, 所以我们只要在一个回文数的前面加上 `%0c` 即 `/f` 即可, 得到 flag:



find a strange dongxi: hctf{go0d_job_intv4l_iz_g00d}

Flag: hctf{go0d_job_intv4l_iz_g00d}

Web.26:不可能拿到的 flag


仔细观察题目:


```
lude_once("flag.php");


(isset($_POST['name']) and isset($_POST['password'])) {
    if ($_POST['name'] == $_POST['password']){
        print 'Your password can not be your name.';
    }else if (sha1($_POST['name']) == sha1($_POST['password'])) {
        die('Flag: '.$flag);
    }else{
        print 'Invalid password';
    }
}
```

前面是双等，后面是三等，查定义加查语法得构造数组 name[]=1&&password[]=111 即可:

INT SQL XSS Encryption Encoding Other

 Load URL

 Split URL

 Execute

☒ Enable Post data ☐ Enable Referrer

Post data


Flag: `hctf{o0k!!g3t_f14g_s0_ez}`


flag 为: `hctf{o0k!!g3t_f14g_s0_ez}`


`web.27.php` 真可怕我要回农村

根据题目, 要满足 `is_array($b)`, `!is_numeric($b)`, `(int)(($a + $b) * 10) == "8"`, 构造 `b=.74asd` 即可:

INT SQL XSS Encryption Encoding Other

 Load URL

 Split URL

 Execute

☒ Enable Post data ☐ Enable Referrer

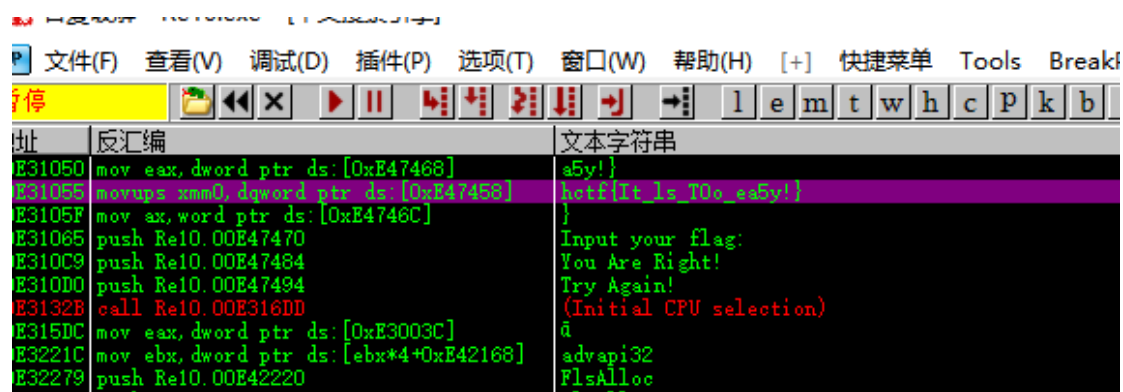
Post data

`hctf{wochubuxiaque_over}`

Flag: `hctf{wochubuxiaque_over}`

Re.28:你看看, 逆向多简单!

OD 载入直接查找字符串即可:



Flag: hctf{It_Is_T0o_ea5y!}

Re.29:蛤，这是啥？

首先 pyc 解密:网址: <http://tool.lu/pyc/>



得到代码：“

```
def what_the_fuck(a):
    if not isinstance(a, str):#a 不是字符串
        raise TypeError
    haihiahia = {0: 'A',1: 'B',2: 'C',3: 'D',4: 'e',5: 'f',
6: 'g',
7: 'h',
8: 'l',
9: 'j',
10: 'K',
11: 'L',
12: 'm',
13: 'n',
14: 'o',
```

```
15: 'p',
16: 'Q',
17: 'R',
18: 'S',
19: 'T',
20: 'u',
21: 'v',
22: 'w',
23: 'x',
24: 'Y',
25: 'Z',
26: '2',
27: '3',
28: '4',
29: '5',
30: '6',
31: '7' }
```

```
str_len_mod5 = len(a) % 5 #a 为 input_str
```

```
bin_of_str = ""
```

```
for c in a:
```

```
    bin_of_chr = bin(ord(c))[2:]
```

```
    length = len(bin_of_chr)
```

```
    bin_of_str += '0' * (8 - len(bin(ord(c))[2:])) + bin(ord(c))[2:]
```

```
if str_len_mod5 == 1:
```

```
    extra_zero = '00'
```

```
    extra_equal = '====='
```

```
elif str_len_mod5 == 2:
```

```
    extra_zero = '0000'
```

```
    extra_equal = '===='
```

```
elif str_len_mod5 == 3:
```

```
    extra_zero = '0'
```

```
    extra_equal = '=== '
```

```
elif str_len_mod5 == 4:
```

```
    extra_zero = '000'
```

```
    extra_equal = '= '
```

```
bin_of_str += extra_zero
```

```
continue
```

```
five_slice = [ bin_of_str[i:i + 5] for i in range(0, len(bin_of_str), 5) ]#0,5,10,15
```

```
output = ""
```

```
for outchar in five_slice:
```

```
    alplabet_ord = int(outchar, 2)#2 转 10
```

```
    output += haihia[alplabet_ord]
```

```
output += extra_equal
return output
```

```
input = raw_input('Your Input:')
print what_the_fuck(input)
”
```

是魔改的 base32,但是看他的编号没有修改,所以很简单,直接把 code 小写转大写:
NBRXIZT3MJQXGZK7GMZCC7I= , 然后解密 (比如在线网站:
http://tomeko.net/online_tools/base32.php?lang=en), 得 flag: hctf{base_32!}

Pwn.24:pwn step0

直接 ida 载入, 看到:

```
push    0                ; buf
push    eax              ; stream
call    _setbuf
add     esp, 10h
sub     esp, 0Ch
push    offset s         ; "so, can you find flag?"
call    _puts
add     esp, 10h
sub     esp, 0Ch
push    12345678h
call    foo
add     esp, 10h
mov     eax, 0
mov     ecx, [ebp+var_4]
leave
lea     esp, [ecx-4]
retn
main endp
```

```

add     esp, 10h
sub     esp, 0Ch
lea     eax, [ebp+s]
push    eax                ; s
call    _puts
add     esp, 10h
cmp     [ebp+arg_0], 'aaaa'
jnz     short loc_80485D1

```

```

call    getFlag

```

```

loc_80485D1:
nop
leave
retn
foo endp

```

12345678h 本来是不可能等于'aaaa' (61616161h) 的，但上面用的是 gets，所以我们可以用 gets 构造栈溢出，把之前 push 进来的 12345678h 覆盖成 aaaa，计算两者中间的 push，pop 和 call，需要长度为 50（可能算错，及少于 50）的 a 字符串。nc 121.42.25.113 10000 后打入长度 50 的 a 得到 flag:

```

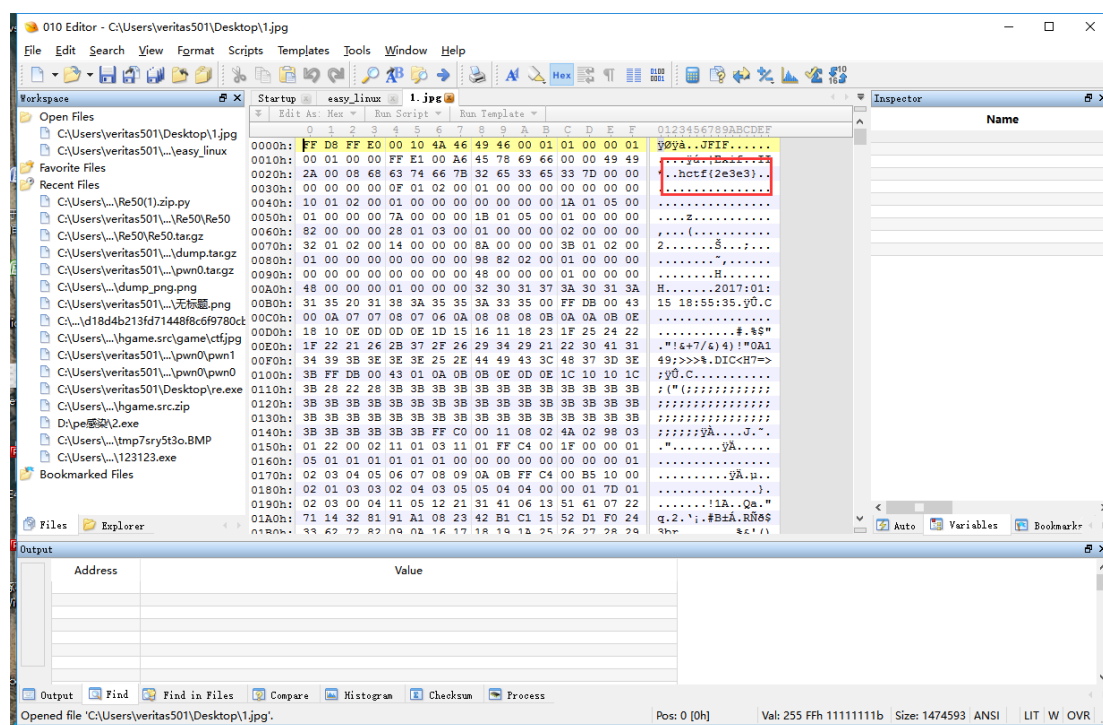
so, can you find flag?
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
rctf{Pwn_1s_1nteRestIng}
ver@debian:~/下载$ █

```

Flag:hctf{Pwn_1s_1nteRestIng}

Misc.32:Explorer 的图库之一:

不解释:



Flag: hctf{2e3e3}

Misc.31:Explorer 的图库之二：

先打开 binwalk 扫一下：

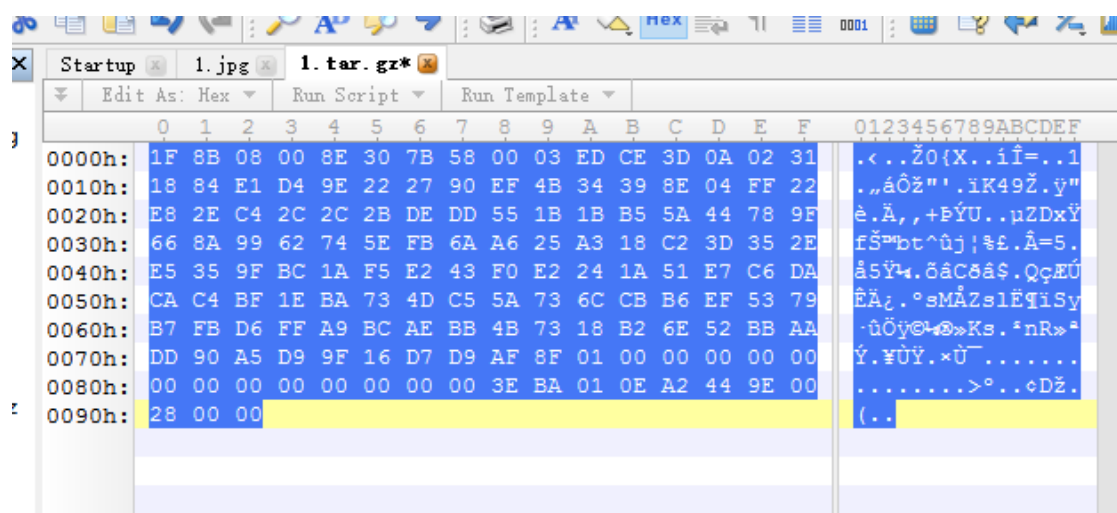
```

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
ver@debian:~/下载$ binwalk 1.jpg

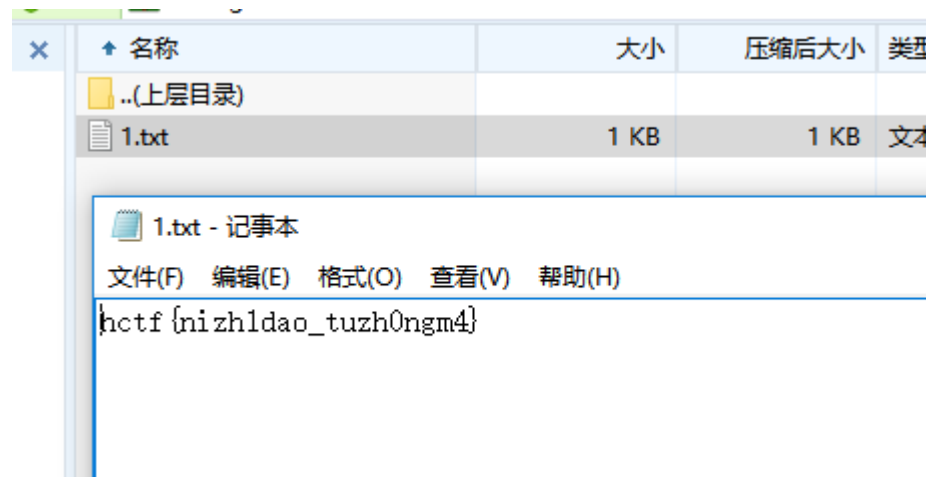
DECIMAL          HEXADECIMAL      DESCRIPTION
-----
0                0x0              JPEG image data, JFIF standard 1.01
45654            0xB256           gzip compressed data, from Unix, last modified: 20
17-01-15 08:19:26
45801            0xB2E9           PNG image, 1500 x 1072, 8-bit/color RGB, non-inter
laced
45842            0xB312           Zlib compressed data, default compression

```

提取中间的 tar.gz



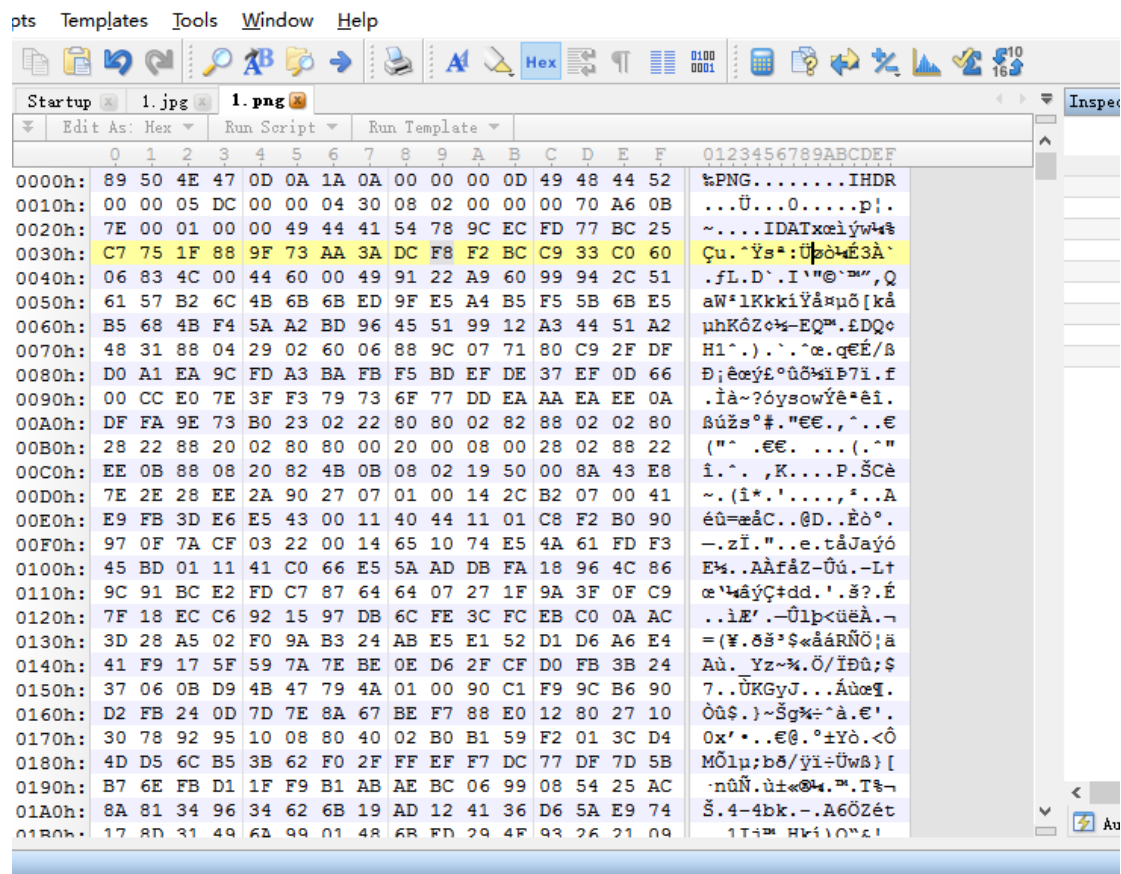
解压得:



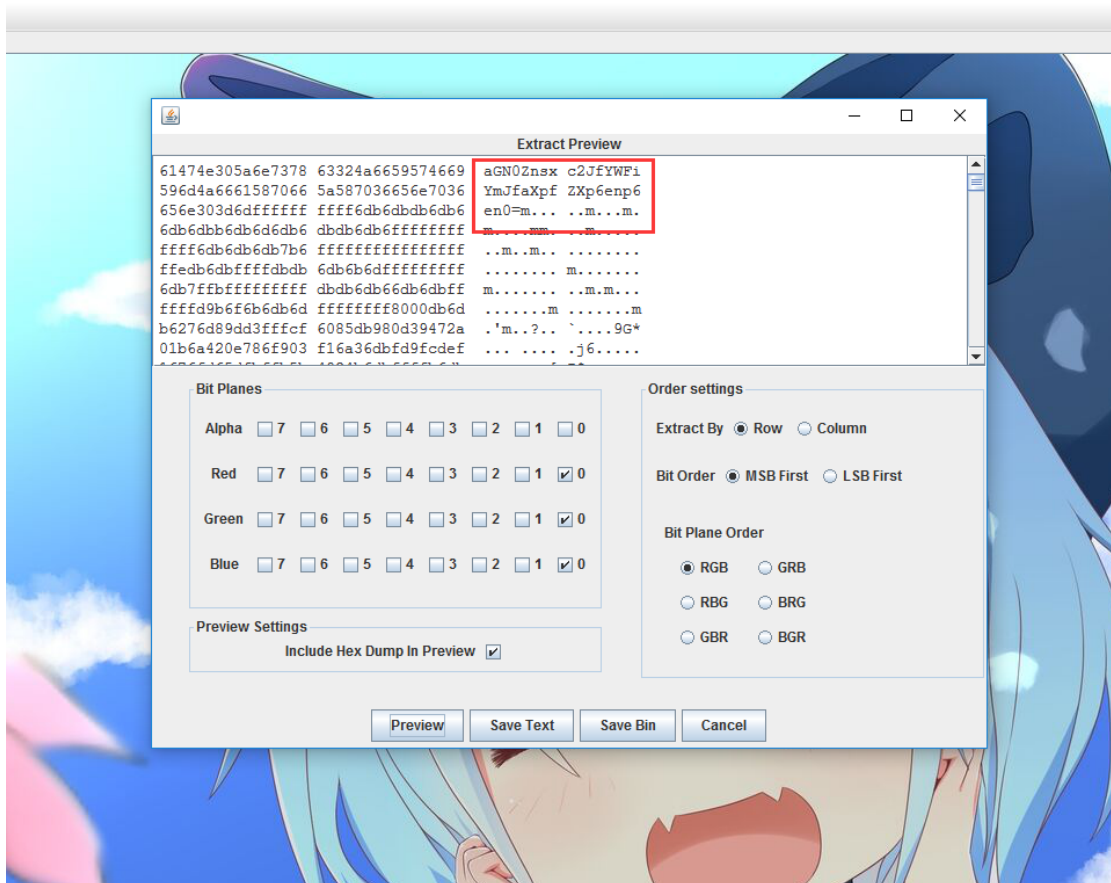
Flag: hctf{nizh1dao_tuzh0ngm4}

Misc.30:Explorer 的图库之三:

同 31 的方法，再提取后面的 png:



，取出来以后用 stegsolve.jar 查看



发现可疑字符串 aGN0Znsxc2JfYWFiYmJfaXpfZXp6enp6en0= , base64 解密得 flag : hctf{1sb_aabbb_iz_ezzzzzz}

Misc.38:explore 的奇怪番外 1:

这题我解得很奇怪就没啥 wp，就是麒麟臂，使劲狂点，点出 flag（现在不想再点第二次，故无法提供 flag）

Crypto.18:密码学入门（一）:

p:

0x9a724c6747de9eadccd33f4d60ada91754b8be8c65590cafe66f69a2f4afbfd359e47ca6fd2dbde8948062dc116bc574f4313ab99b2bb6d8ae47beaa0c1ebeddL

q:

0x8c1c81cc005ce3dd6d684ebb88151dc0c53b1cef8a29b1cb8121860fb57d93117bf449aac4300dc6103ac6211c6f8ae68987d99aff0dd8967a4afa00f2116873L

e:

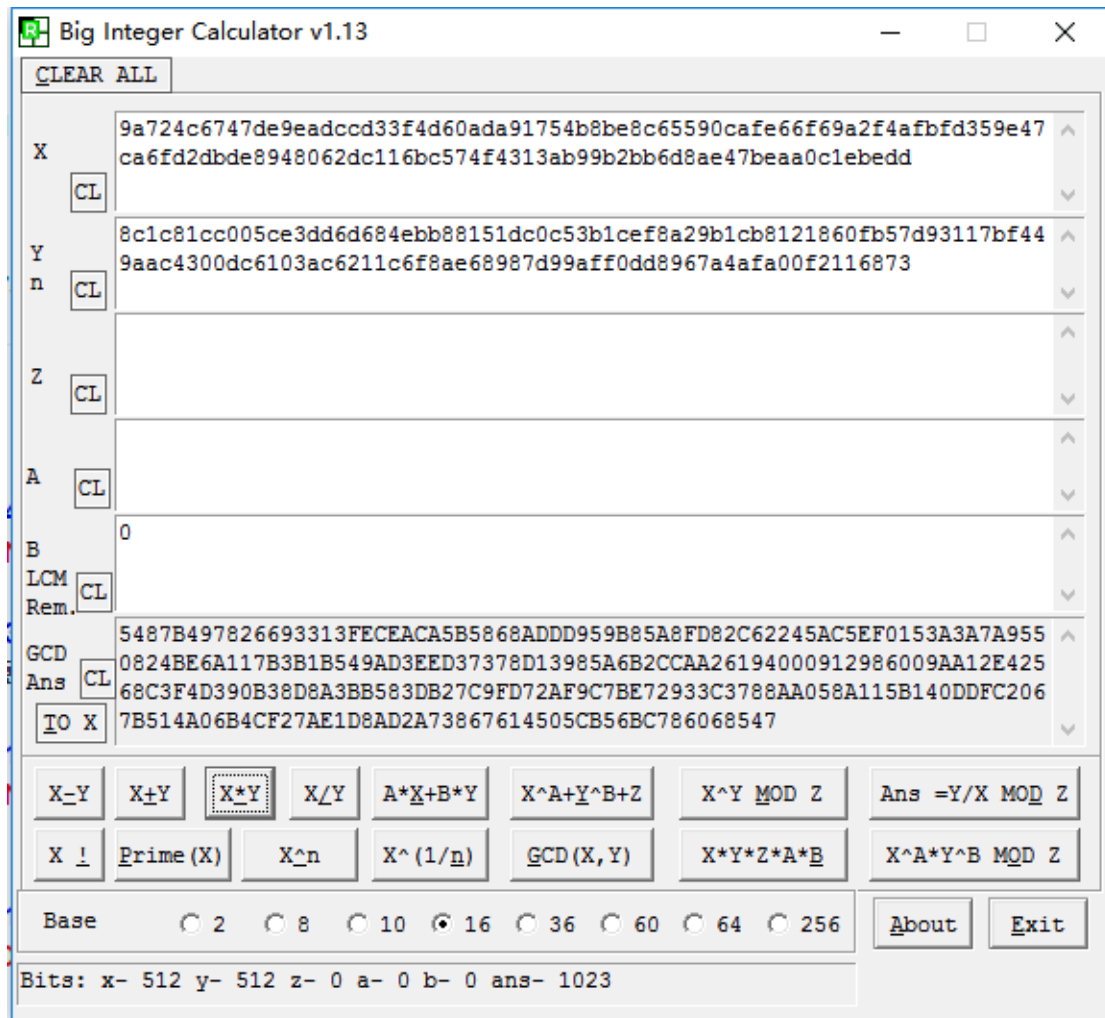
0x190a000845e9c8c2059242835432326369aaf8c7ca85e685bba968b386155a91f1f7ca1019ff23d119222e1f0dfdeb0915d2e97601ef94bf15ca6d9211e984e9038f263f4984355c397ed22d67c26da6d31acfc4d599c70cba80859bee099e5a2dc3ab23aecf58f73f44d07318f70985c623d9612efefb15bf8dab77d5d54e85L

d: 0x28b95b7e3159a851cbf537e007ae49864b7dbb93fc370a5L

c:

0x23091e42fa7609c73f1941b320fad6d2ff6e47be588d1623f970f1fee7abd221c9834b208f3c8889
02fe87ca76ec1e1363757d93c6e25c49f1c61c72b141c0b8848b54a117427d8e30eeab89694eb5f8
49cafecb0e5361b9b2b0e3f89e0fdbcc66a6aad4a1a4a85d828083a01a5d569b7eeb6f9151794453
382b524aa52993f9L

由 $m=c^d \bmod n, n=p*q$ 知, 用大数计算器解得



n0x5487B497826693313FECEACA5B5868ADDD959B85A8FD82C62245AC5EF0153A3A7A9550824
BE6A117B3B1B549AD3EED37378D13985A6B2CCAA26194000912986009AA12E42568C3F4D390
B38D8A3BB583DB27C9FD72AF9C7BE72933C3788AA058A115B140DDFC2067B514A06B4CF27AE
1D8AD2A73867614505CB56BC786068547

m=6867616D657B7273615F31735F763372795F65347379217D

6867616D657B7273615F31735F763372795F65347379217D

16进制转字符 字符转16进制 清空结果

hgame(rsa_1s_v3ry_e4sy!)

Flag: hgame(rsa_1s_v3ry_e4sy!)

Crypto.19:密码学教室入门（二）

简单的凯撒，写了个脚本：

```
a='mlfrj{Hfjfw_hnumjw_8x_ozxy_ktw_kzs}'
```

```
for i in range(27):
```

```
    b = []
```

```
    for c in a:
```

```
        if c.isalpha() :
```

```
            c = chr(ord(c)-1)
```

```
            if c.isalpha() == 0:
```

```
                c = chr(ord(c)+26)
```

```
        if c.isdigit():
```

```
            c = chr(ord(c)-1)
```

```
            if c.isdigit() == 0:
```

```
                c = chr(ord(c)+10)
```

```
        b.append(c)
```

```
a = ''.join(b)
```

```
print(a)
```

得到 flag: hgame{Caesar_cipher_3s_just_for_fun}，但是提交后不对，分析得把 3s 改为 1s 提交成功，最后的 flag: hgame{Caesar_cipher_1s_just_for_fun}

Crypto.20: 密码学教室入门（三）

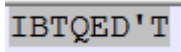
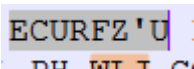
首先除空格：

ET. PESFVWI, AJGZII BU D LJSQ ISW IKV MRQTLWTOOHRY JP WLJ CCVXNMNH, XJTVLJNFU RR IBTQED'T DHLFMH DX MJU WVNBN. GEWO CB MX SGOIFTGG, SSMA WS GF CUVJTVHH FHCLR QBVHV YICW HFZ. C QIB UTLEQ CGJMST QQ XMF HRPQPYLRL ECB, YSEGU RJX EKEWHGV FWPWJLY CA WLJ EGIEWHGV ESE C WLNSF LRIJXLHZN ZLT JU VSTO THZJBNHH FT FU. QFOGWXJ. IG KEI XTLXYFP DR FDERYSU QI LNT KPTWJURRRFPW EY UJH LFOFV SK ECURFZ'U IEYIGU ESE JLIHIF LX NO JLW HFNO; HJGCUKJ GQXRI JV ZLNMG VIFSEKMSH VKI HFNO HZSKQK YIG VXTSOLRL PH WLJ CCVXNMNH.

ETPESFVWIAJGZIIIBUDLJSQISWIKVMRQTLWTOOHRYJPWLJCCVXNMNHXJTVLJNFURRIBTQEDTDHLMHDXMJUWVNBNGEWO CBMXSGOIFTGGSSM AWSGFCUVJTVHHFHCLRQBVHVYICWHFZCQIBUTLEQCGJMSTQQXMFHRRPQPYLRLCEBYSEGURJXEKEWHGVFWPWPWJLYCAWLJEGIEWHGV ESE C WLNSF LRIJXLHZNZLTJUUVSTO THZJBNHHFTFUQFOGWXJIGKEIXTLXYFPDRFDERYSUQILNTPWJURRRFPWEYUJHLFOFVSKECURFZUIEYIGUESEJLIHIFLXNOJLWHFNOHJGCUKJGQXRIJVZLNMGVIFSEKMSHVKIHFNOHZSKQKYIGVXTSOLRLPHWLJCCVXNMNH

分析这些染色的单词，可以发现他们相隔 35,150,345,110 的距离，大胆推测 key 的长度为 5，分组得：

EFBJSIQJCMTFBTMBJBOSTMFTHBIZUCTFPEEXHPCEHESJBJOBTOIXFDUTJFUOEZIEFOFGGJMSHFSISPCM
TVGUQKTOPCNVUTDHUNCGGACVCVCTGQHYCGEGWAGGCFXNUTNFGGTPEQKUPJFCUGJPJNCQVGEVNGGOHCN
PWZDIVLHWVHLRQHDWGBOGWUHLHWQLJQLBUKJVWVWLLZVHHUWKLDRIPRWHVUIULLLOUXZVKKOQVLVWVH
EILSMWRLXXJRELXVEMISSVHRVHIEMXPRYREFLLEELRHLSZHQEXRYLTRELSREEHXWHKRLIMIHKXRLX
SAIWRRTYJNJNIDFMNWXFSGJFYFBQSMQLSJWWYJWSNIZTTJFFJIYFSNWRYFKFYSINHHJJNFHSZYTJLN

由  处的 T 一定是 s（英文中'前这么长而且后面跟着字母的只有 s 了吧）以及  中的 u 是 s 知道，一二两条解密得：

DEIARHPNIBLSEASLIANRSLESGAHTBSEODDWGOBBDGDRIAINASNHWECTSIETNDYHDENEFFILRGERHROBL
RTESOIRMNALTSRBFSLAEYATATAAREOFWAECEUYEEADVLSRLDEERNCOISNHDASEHNHLAOTECTLIEMFAL
从而开头的 ET.对应为 DR.印证了我们的假设。在根据 ESE 的后一个 E 在第一条中对应为 D 猜测 ESE 为 AND，从而解出了 4,5 两条：

AEEHOISNHTTFNAHTRAIEOORDNRDEAITLNUNABHHAHNDHOVDMTATNUHPNAHONAADTS DGNHEIEDGTNHT
NVDERMOTIEIDYAHIRSANBEALTAWLNHLGNERRTERNIDUOOEAAEDTANIRMTAFATNDICEEDIANCUTOGEI

发现此时的 WLJ 的 J 对应 HE,猜测 WLJ 对应 THE，从而解得第三条：

MTWAFSIETSEIONEATDYLDTREIETNIGNOIYRHSGTFTSIWSEERTHIAOFMOTESRFRIILRUWSSHLSITSE

所以开头为：DR. MANETTE, VIEWED AS A HERO。。。百度搜索得：

[A Tale of Two Cities - Wikipedia, the free encyclopedia](#)



查看此网页的中文翻译，请点击 [翻译此页](#)

A year and three months pass, and Darnay is finally tried.Dr.

Manette, viewed as a hero for his imprisonment in the ha...

[en.wikipedia.org/wiki/...](https://en.wikipedia.org/wiki/A_Tale_of_Two_Cities) - [百度快照](#)

得 flag: hgame{A_Tale_of_Two_Cities}

Crypto.33: 密码学教室入门（四）

这道题其实用不到 rsa，由 16 进制敏感可以发现，c:的前几位为 686761h，明显对应 ascii 码，直接解密得：flag: hgame{rsa_1s_still_e4sy_now!}

Crypto.37:密码学教室番外篇

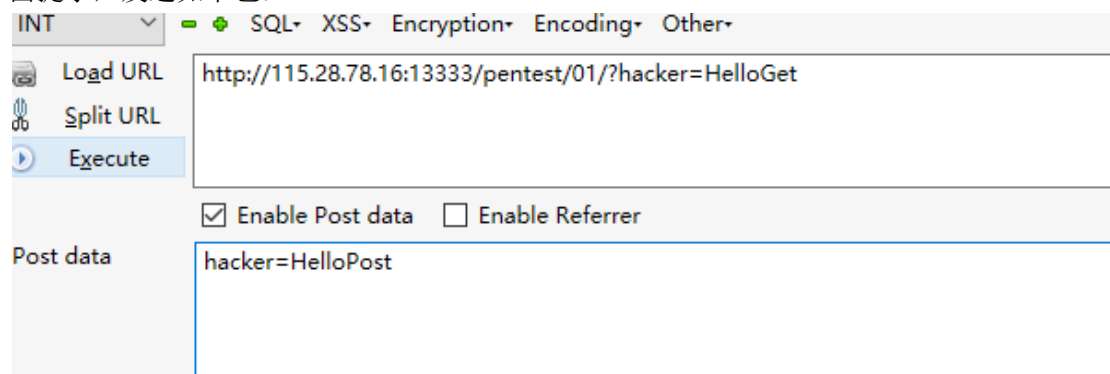
同 19，脚本解得：

```
jicog{fihmujctd04103437033//+/%}  
ihbnf{ehgezidsz53398346524//+/%}  
hgame{dgfdyhcry42287235413//+/%}  
gfhld{cfecxgbqx31176124302//+/%}  
feykc{bedbwfapw20065013291//+/%}  
edxjb{adcavezov19954902180//+/%}  
dcwia{zcbzudynu08843891079//+/%}  
cbvhz{ybaytcxmt97732780968//+/%}
```

Flag: hgame{dgfdyhcry42287235413//+/%}

Pentest.14: lightless 的渗透教室入门篇（一）

由提示，发送如下包：



Flag 可见：

题目内容：

- 向本页面同时发送GET和POST请求；
 - GET请求内容为hacker=HelloGet
 - POST请求内容为hacker=HelloPost
- 如果你不知道如何发送POST请求，方法一：学习curl命令。方法二：学习burp工具。方法三：学习Chrome/Firefox上的开发者工具或各种浏览器插件。

hctf{PostAndGetIsSoEasy_comeon!}

Flag: hctf{PostAndGetIsSoEasy_comeon!}

Pentest.16: lightless 的渗透教室入门篇（三）

先抓包，发现如下字段：

Varv Accept-Encoding
cookiecontent admin=1 and isLogin=true
x-powered-by PHP/5.4.41

请求头信息

原始头信息

Accept text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding gzip, deflate
Accept-Language zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Cache-Control max-age=0
Connection keep-alive
Host 115.28.78.16:13333

Upgrade-Insecure-Requests 1

修改 cookies:

Name:	Cookie	Value:	admin=1;isLogin=true	Add
Name	Value			
Cookie	admin=1;isLogin=true	Delete		

本次内容十分重要，请重点掌握，以后XSS时也会用到这部分内容。

- [什么是cookie](#)
- [我知道你们没人看资料](#)
- [啥是session](#)
- [cookie和session的区别](#)

- 如果你不想看英文的资料，自己去搜中文的看吧。

(-_-) 你们真的而有人看这书么？扩展阅读：《HTTP权威指南》，经典必读书目，drops曾经有提到过。

题目内容：

- 设置cookie，内容自行寻找，都是以前学过的内容。
- 如果你不知道如何伪造cookie，方法一：学习curl命令。方法二：学习burp工具。方法三：学习Chrome/Firefox上的开发者工具或各种浏览器插件。

flag: hctf{hao_hao_kan_zi_liao!!!} 通过伪造cookie，你可以绕过一些限制，或是伪造身份。

HEADERS

Flag: hctf{hao_hao_kan_zi_liao!!!}