《网络攻防实战》实验报告

第 8 次实验： RE

姓名： 罗嘉璐

学号： 211220047

21 级 计算机科学与技术 系

邮箱： 211220047@smail.nju.edu.cn

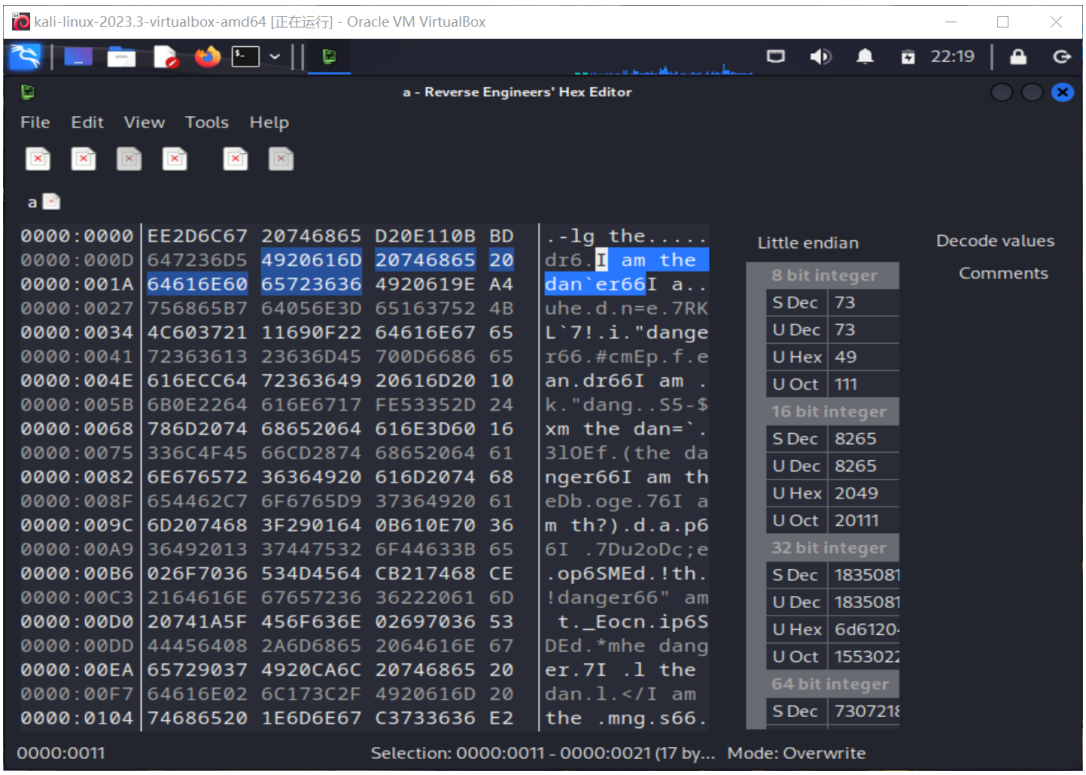
时间：2023.11.19

一、实验目的

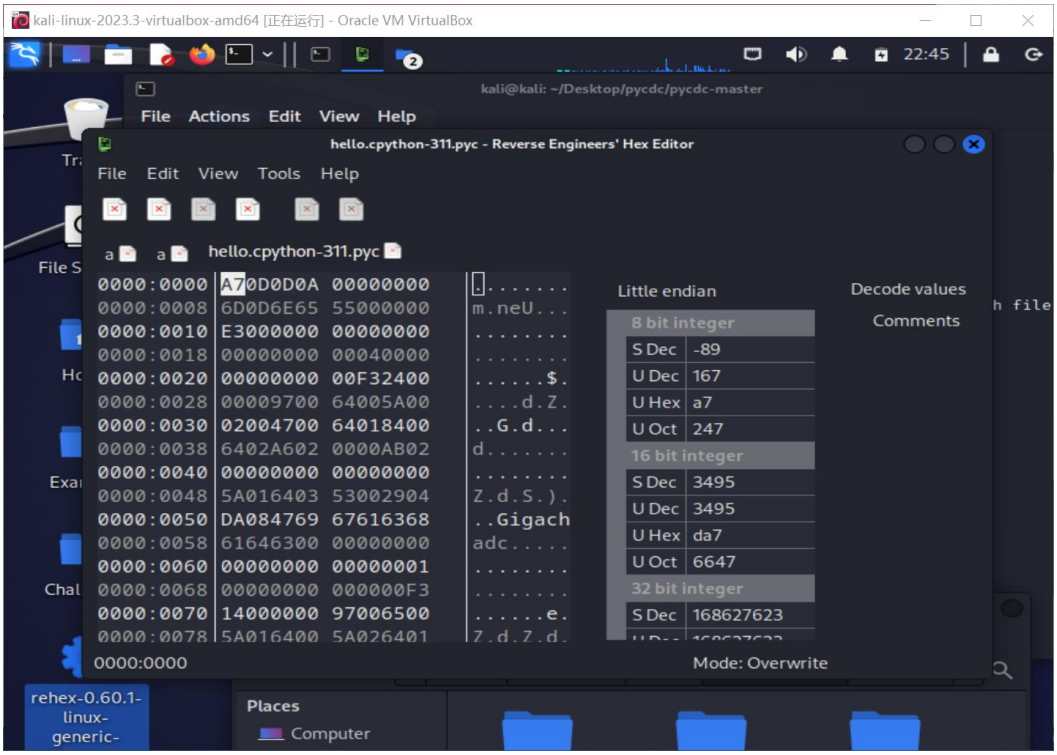
通过分析一个奇怪的二进制文件获得 flag。

二、实验内容

1、 这个二进制文件是.pyc 和一个 OTP 字符串异或得到的。用 rehex 打开二进制文件 a。



观察右边字符串发现很多重复的 I am the danger66 类似串。然后二进制打开 hello.pyc 文件，I am the danger66 对应位置很多是 0，通过异或的原理猜测 OTP 字符串是 I am the danger66 的 16 进制码。

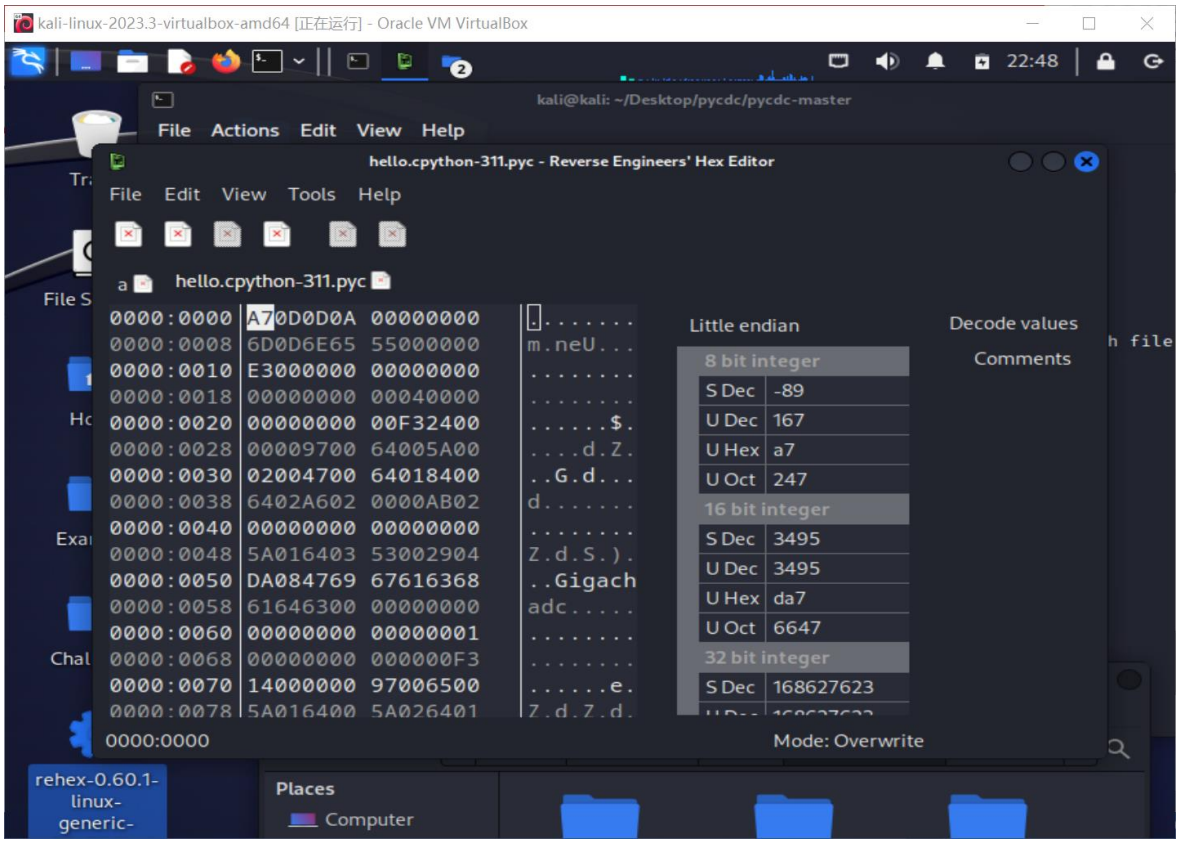


解密脚本：

```
def xor_hex_strings(hex_str1, hex_str2):
    int_val1 = int(hex_str1, 16)
    int_val2 = int(hex_str2, 16)
    xor_result = int_val1 ^ int_val2
    return format(xor_result, '032x')

# 读取文件"a"的内容
with open("a", "rb") as file:
    file_content = file.read()

original_hex = ''.join(format(byte, '02x') for byte in file_content)
print("原始文件内容（按字节）:")
print(original_hex)
def xor_hex_strings(hex_str1, hex_str2):
    int1 = int(hex_str1, 16)
    int2 = int(hex_str2, 16)
    result = int1 ^ int2
    return format(result, 'x')

def xor_loop(hex_input, hex_key):
    result = ""
    j = -1
    for i in range(0, len(hex_input), 2):
        hex_byte = hex_input[i:i+2]
        j=(j+1)%len(hex_key)
        key_byte = hex_key[j]
        j=(j+1)%len(hex_key)
        key_byte+=hex_key[j]
        if j==31:
            a=1
            b=1
        tmp = xor_hex_strings(hex_byte, key_byte)
        if len(tmp) == 1:
            tmp = '0' + tmp
        result += tmp
        result+=" "
    return result

# 输入的内容
input_hex =original_hex
key_hex = "4920616d2074686520646616e6765723636"

# 进行异或操作
```

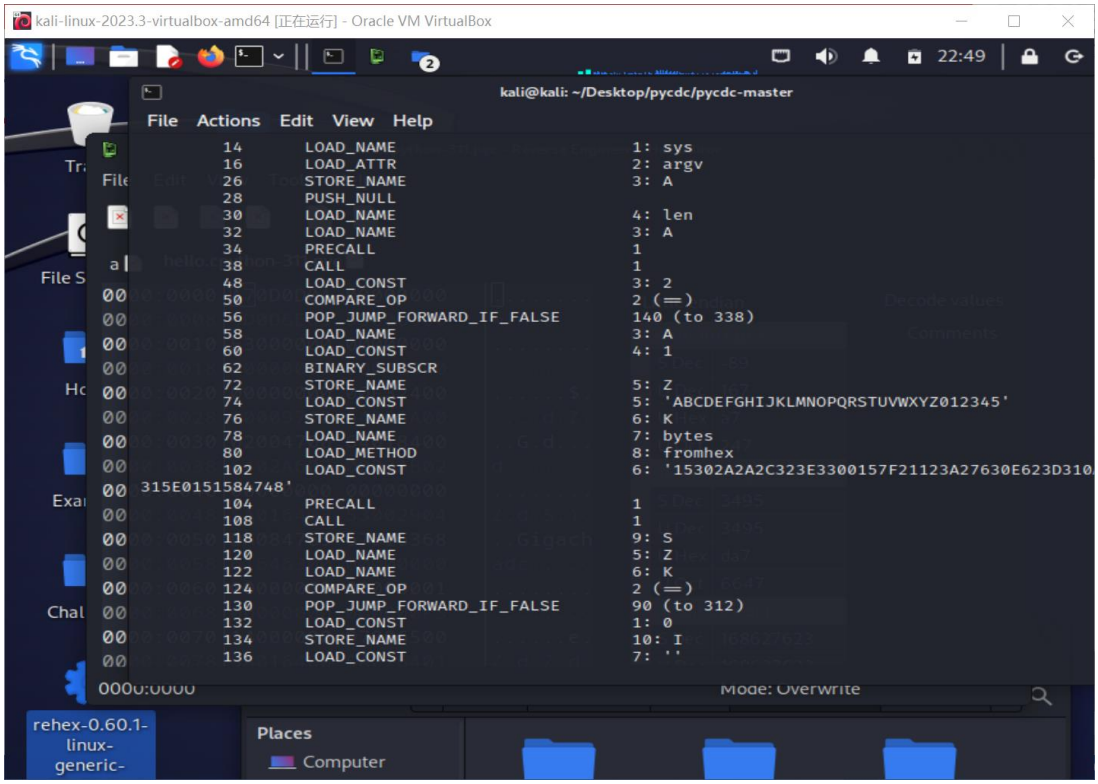
```
result_hex = xor_loop(input_hex, key_hex)

# 输出结果
print(result_hex)
```

将解密后的 16 进制的 2 进制码进行替换，然后运行 python a，提示要输入密码




```
┌──(kali㉿kali)-[~/Desktop/pycdc/pycdc-master]
└─$ ./pycdc a
# Source Generated with Decompyle++
# File: a (Python 3.11)

Unsupported opcode: POP_JUMP_BACKWARD_IF_TRUE
__doc__ = '\nChallenge A\n'
import sys
A = sys.argv
# WARNING: Decompyle incomplete
```

```
┌──(kali㉿kali)-[~/Desktop/pycdc/pycdc-master]
└─$ ./pycdas a
a (Python 3.11)
[Code]
    File Name: /home/yang/Seafile/网络攻防实战/逆向/课件A/Challenge/src/main.py
    Object Name: <module>
    Qualified Name: <module>
    Arg Count: 0
    Pos Only Arg Count: 0
    KW Only Arg Count: 0
    Stack Size: 7
    Flags: 0x00000000
    [Names]
        '__doc__'
        'sys'
        'argv'
        'A'
        'len'
        'Z'
        'K'
        'bytes'
        'fromhex'
        'S'
        'I'
        'F'
        'chr'
```



通过 pycdas 查看密码，然后运行 a，获得 flag：Trinity{I_4m_th3_0ne_wh0_kn0cks}

```
┌──(kali㉿kali)-[~/Desktop/pycdc/pycdc-master]
└─$ python a "ABCDEFGHIJKLMNOPQRSTUVWXYZ012345"
Trinity{I_4m_th3_0ne_wh0_kn0cks}
```

思考过程：尝试重写 marshal.loads 绕过 OTP 解密，但是失败了，然后开始观察二进制文件，观察到二进制文件的 16 进制形式对应的字符串有规律，然后尝试破解 a 文件，最后成功获得 flag。有趣的实验。