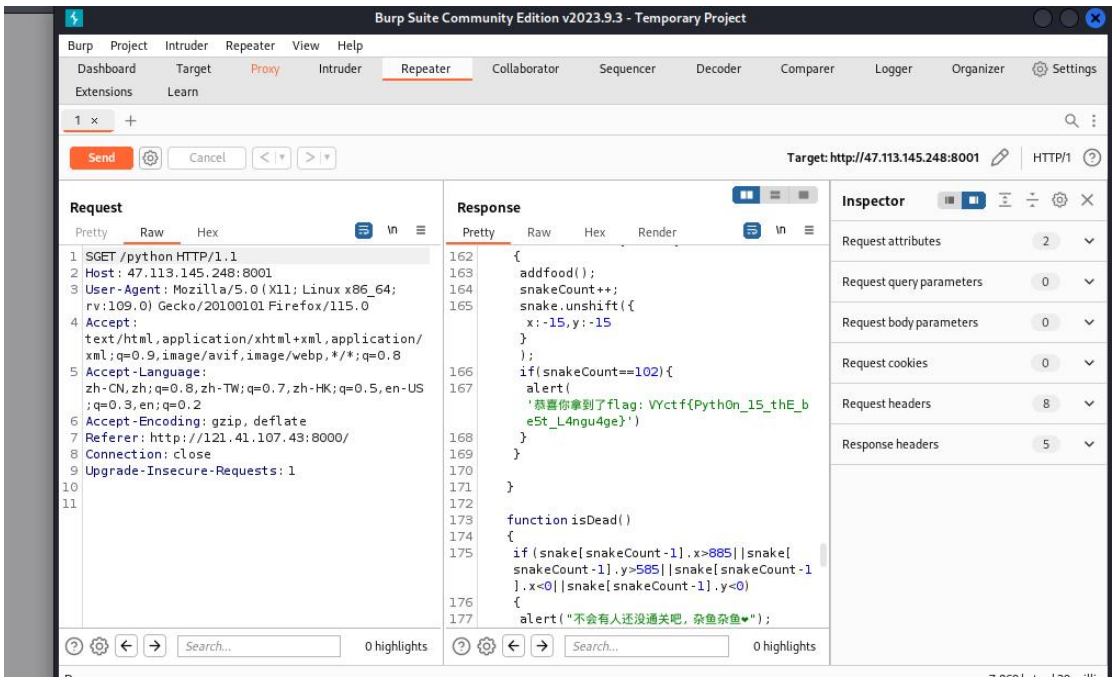
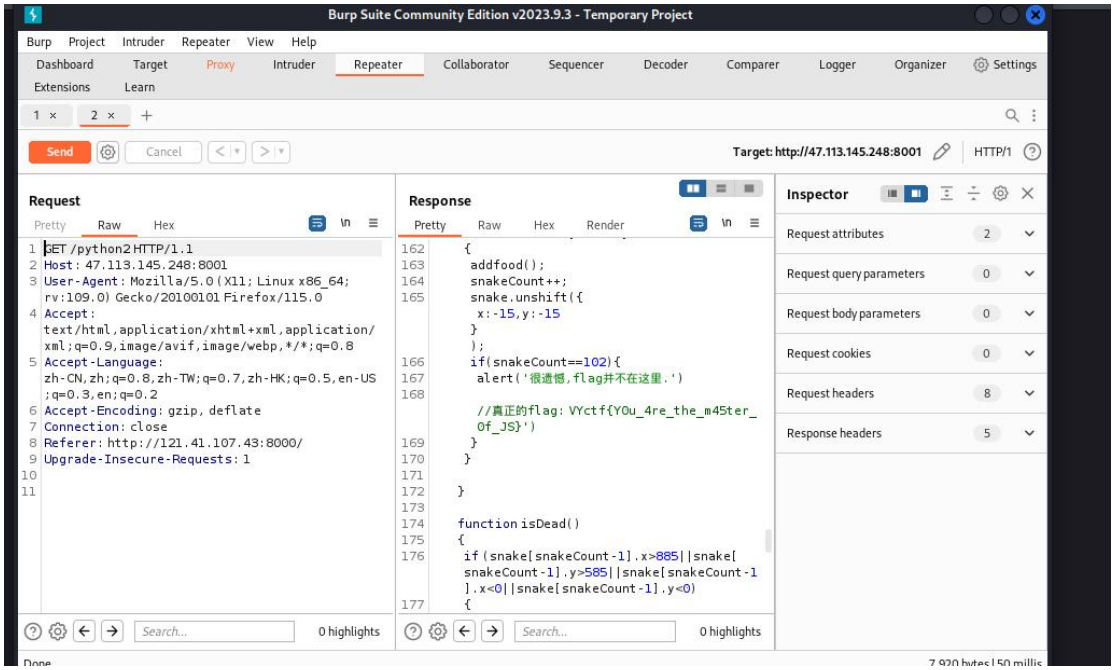
玩蛇

1.方法以：正常玩，当蛇长度足够时即可拿到 flag.
2.方法二：进入网页，利用 burp suite 抓包，发送到重放模块（Repeater）,点击 send,即可在源码中查找到 flag。



玩蛇 2.0

依据以上第二种方法，拿到 flag。



玩具沙盒

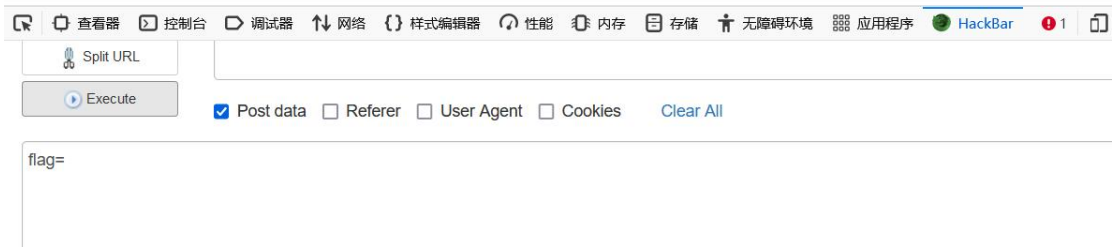
1.对目标源码进行审计，发现需要满足特定条件才能获取 flag。

main.py box.c

```
19
20      read(0, &filt_len, sizeof(uint32_t));
21      uint8_t *filt =  (unsigned char *)calloc(sizeof(uint8_t), filt_len);
22      int res = read(0, filt, filt_len);
23      if (res != filt_len) {
24          printf("太多了读不完了555");
25          return 1;
26      }
27
28      if (install_seccomp(filt, (unsigned short)filt_len))
29          return 1;
30
31
32      return 0;
33  }
34
35  int install_seccomp(unsigned char *filt, unsigned short filt_len) {
36      struct prog {
37          unsigned short len;
38          unsigned char *filt;
39      } rule = {
40          .len = filt_len >> 3,
41          .filt = filt
42      };
43      if (prctl(PR_SET_NO_NEW_PRIVS, 1, 0, 0, 0) < 0) {
44          printf("滴滴，安全管制!");
45           return 1;
46      }
47
48      printf("干得漂亮! flag是vyctf{test_flag}");
49
50      return 0;
```

2.使用火狐浏览器插件 HackBar 传入特定的 Post 参数，拿到 flag。（其实是一步步试探，最后没注意忘记粘贴参数点运行就出来了，代码都还没理解完全）

干得漂亮! flag是vyctf{th1s_is_c0de9ate_baby_b0x}



这亦是一种图片

1.根据题目，进入 kali,利用 xxd 命令运行即可得到 flag，忘截图了

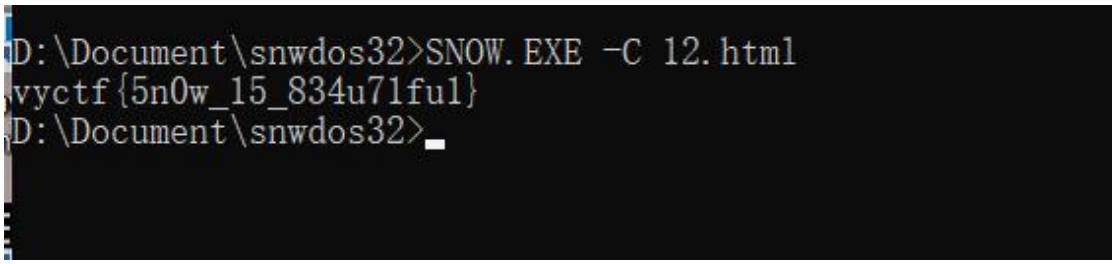
snow(雪)

1.查看源码，审计后发现好像没什么发现，看题目后猜测使用了 snow 隐写，仔细查看源码发现确实存在：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Snowflake Effect with Text</title>
    <style>
        body {
            overflow: hidden;
            background-color: #333;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            font-family: 'cursive', sans-serif;
            color: white;
            font-size: 2em;
        }

        .snowflake {
            position: absolute;
```

2.利用 snow 解码工具解码即可得到 flag



简单 ino

1.下载源码，查询到.ino 文件需要特定软件才能打开运行。由于安装了半天的软件都不能正常运行，于是利用万能的记事本打开查看源码（记事本战神！）：

flag.ino - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
// lcd1602:SCL is uno:A5, lcd1602:SDA is uno:A4, lcd1602:VCC is num:V5, lcd1602:GND is uno:GND.

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 20, 4);

int flag[20] = {118, 121, 995, 116, 102, 123, 104, 101, 492, 108, 482, 95, 65, 114, 100, 117, 493, 110, 482, 125};
int line[20] = {10, 3, 14, 4, 0, 13, 10, 3, 14, 0, 14, 0, 0, 7, 13, 5, 14, 0, 14, 7};
int i = 0;

void setup() {
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Hello VYctf!");
}

void loop() {
  delay(1000);
  lcd.clear();
  lcd.print("flag is:");
  lcd.setCursor(line[i], 1);
  lcd.print(flag[i]);
  i++;
}
```

第 1 行，第 1 列 100% Windows (CRLF) UTF-8

2.简单进行代码审计后，猜测 flag[20]内是真正 flag 的 ascii 编码。打开 pycharm，简单利用前面几个进行 ascii 解码，发现 118 是'v'，121 是'y'，但是'c'是 99 而不是 995，于是将原 flag 稍加修改，构造新的 flag1，编码后得到真正的 flag

1.py

```
flag = [118, 121, 995, 116, 102, 123, 104, 101, 492, 108, 482, 95, 65, 114, 100, 117, 493, 110, 482, 125]
line = [10, 3, 14, 4, 0, 13, 10, 3, 14, 0, 14, 0, 0, 7, 13, 5, 14, 0, 14, 7]

flag1=[118, 121, 99, 116, 102, 123, 104, 101, 49, 108, 48, 95, 65, 114, 100, 117, 49, 110, 48, 125]
aa=bb=cc=""

# 118, 121, 995, 116, 102, 123, 104, 101, 492, 108, 482, 95, 65, 114, 100, 117, 493, 110, 482, 125
# 118 121 99 116 102 123 125
print("flag is:")

# for i in line:
#     aa+=chr(flag[i])
# print(aa)
#
# for i in flag:
#     bb+=chr(i)
# print(bb)

for i in flag1:
    cc+=chr(i)
print(cc)
```

运行 1

```
D:\app_2\python-3.11.4\python.exe D:\Document\pythonProject\vyctf\1.py
flag is:
vyctf{he1l0_Ardu1n0}

进程已结束，退出代码为 0
```

还原大师

1.下载源码后放入 pycharm，发现无法正常运行。

```
def function_dict(position):
    datas = [
            "1sd2jk}3l",
            "wurio456{",
            "8cvn_xm79",
            "etyufgh14",
            "sVhjlaewY",
            "fhjl_ebco",
            "Ysucinowe",
            "0nt5bw_fn"
            ]
    for data in datas:
        assert len(data) == 10,"your dictionary is not quite right."

    return datas[position[0]][position[1]]

3 用法
def function_base64(left, right):
    assert left == base64.b64encode(right),"base64 verification failed."
    return left[0] & 7

def main():
    datas = []
    datas.append([function_base64( left: b"d2VsY29tZQ", right: b"welcome"),1])
    datas.append([function_base64( left: b"dG8", right: b"to"),8])
    datas.append([function_base64( left: b"VlljdGY", right: b"VYctf"),4])
    datas += [
            [7,3], [3,4], [1,9],
            [3,2], [7,0], [1,2],
            [2,5], [5,1], [0,0],
function_base64()
运行
```

2.先将初始部分进行修复，再次运行发现依旧报错

```
def main():
    datas = []
    datas.append([function_base64( left: b"d2VsY29tZQ==", right: b"welcome"), 1])
    datas.append([function_base64( left: b"dG8=", right: b"to"), 8])
    datas.append([function_base64( left: b"VlljdGY=", right: b"VYctf"), 4])
    print(datas)
    datas += [
            [7,3], [3,4], [1,9],
            [3,2], [7,0], [1,2],
```

3.猜测 datas 部分仍存在代码缺失，利用 main 函数中的数据进行不断实验，依次探测出 V、Y、c、t、f、{、} 各字符所在位置，将其所在位置进行与其他字符位置进行比对，尝试找出缺失字符所在位置。

```
aa = bb = ""

for i in line:
    a = i[0]
    b = i[1]
    # bb += datas[a][b]
    # if a in [0,1,6,7]:
    #     b-=1
    aa += datas[a][b]

print(aa)
# print(bb)

# 0【0|、3、7】
# 1【|2、3、5、6、9】
#
# 2【|2、4、5、6】
# 3【0、2、4】
# 4【1、2、8】
# 5【|1、5】
#
#
# 6【|4】
# 7【0、|3、7】
```

4.多次进行探测后，找出所有缺失字符所在位置，利用任意字符顶替（缺失字符不会是 flag 中的字符，故使用任意字符占位），运行修复后的文件，得到 flag。

```
datas = [
    "1 sd2jk}3l",
    " wurio456{",
    " 8cvn_xm79",
    "etyufgh14",
    "sVhjlaewY",
    " fhjl_ebco",
    "Ysu cinowe",
    "0n t5bw_fn"
]
line = [
    [4, 1], [4, 8],
    [6, 4], [7, 3],
```

```
运行  2-1
D:\app_2\python-3.11.4\python.exe D:\Document\pythonProject\vyctf\d1ct1on4ry\2-1.py
VYctf{y0u_f1xed_the_d1ct1on4ry}

进程已结束，退出代码为 0
```

素数分解

1.查看题目，提示使用 rsa 算法加密，简单了解 rsa 加密流程后，查询 rsa 解密方法

2. 消息加密:

m^e除以n求余数即为c(密文)

$$m^e \equiv c \pmod{N}$$

3. 消息解密:

c^d除以n求余数即为m(明文)

$$c^d \equiv m \pmod{N}$$

```python
# 简单的rsa加密技术
E = 7
P = 17
Q = 163
N = P * Q
# N = 2771= 17 * 163

phin = (P-1) * (Q-1)
# = 16 * 162 = 2592
D = pow(E, -1, phin)
# = pow(7,-1,2592) = 1111

# print(D)
# D = 1111
# PT = open("D:/Document/pythonProject/vyctf/rsa/flag.ct","w")
# with open("D:/Document/pythonProject/vyctf/rsa/flag.pt","r") as file:
#     for f in file.read():
#         PT.write(chr((ord(f) ** E) % N))
# PT.close()
```

2.基于解密方法构造解密代码，利用 pycharm 运行代码得到 flag。

```python
aa='ȟ▯○▯▯ઞĸƘ▯▯▯8▯eड Kғ8ञɻʮeKडस≏'
bb=''
for i in aa:

    bb+=chr((ord(i) ** 1111) % 2771)
print(bb)

# print(pow(7,-1,2592))
```

```
运行  main

D:\app_2\python-3.11.4\python.exe D:\Document\pythonProject\vyctf\rsa\main.py
vyctf{R5a_1s_M0dern_pA55w0rd}

进程已结束，退出代码为 0
```

小小的也很可爱

1.打开题目，提示使用 elgamal 加密，依然简单了解 elgamal 加密

3. ElGamal公钥加密算法

密钥产生过程：

- 随机选择一个大素数 p 以及两个小于 p 的随机数 g 和 x（ g 是 p 的一个原根）；
- 计算 $y \equiv g^x \bmod p$;
- 以(y， g，p)作为公开密钥，x作为私密密钥。

加密过程：

- 设欲加密明文消息为 M;
- 随机选取一个与 p - 1 互素的整数k;
- 计算$C_1 \equiv g^k \bmod p$，$C_2 \equiv y^k M \bmod p$;
- 密文为$C = (C_1, C_2)$ 。

解密过程：

$$M = \frac{C_2}{C_1^x} \bmod p$$

CSDN @じ凉墨

2.查看源代码尝试基于解密方法构造解密代码

```
P = 487
# D = ?
E1 = 31
# E2 = pow(E1, D, P)
E2 = 168
R = 11

# C1 is:162
# C2 is:þÍ≥ÜSOHÍSIWÎ`¦`W?5Y?`É?`Yo¦¦MWÜSHY5≥?MWÉ≥SHYoÖÜ5ÉÉ?`WŔÉSHY5ÜÀ¦ACKÜ

1 个用法
def enc(PT, E1, E2, R, P):
    C1 = pow(E1, R, P)
    C2 = ""
    for i in PT:
        data = (i * pow(E2, R)) % P
        C2 += chr(data)
    return C1,C2

with open("./flag.pt","rb") as PT:
    C1,C2 = enc(PT.read(), E1, E2, R, P)

with open("./flag.ct","w") as CT:
    CT.write("C1 is:"+str(C1)+"\nC2 is:"+C2)

print("C1 is:"+str(C1)+"\nC2 is:"+C2)
```

3.正常方法：在了解 elgamal 加解密原理后，想必能完整构造出解出方法

```python
import math
import numpy as np

result = np.mod( *args: 421 / (162 ** 58), 487)
print(result)



P = 487 # 大素数 P
# D = ?
# D = 58,301,544,787,1030,1273,1516,1759,2002,2245,2488,2731,2974
# 随机数 E1,D
E1 = 31
D = 58
# E2 = pow(E1, D, P)
E2 = 168
R = 11

M = 86
# c1 = pow(E1,R,P)
c1 = 162
c2 = (E2**R*M)%P # (168**11*M)%487
c3 = (c2/pow(c1,D))%P

print(c2)
print(c3)
```

（但我死活就是构造不出解密代码来，老是有错，虽说数字确实很小，嗯，小小的，但就是无法解出来，我怀疑是我的问题。。。于是我就想出了不正常的方法…）

4.不正常方法：正常解题步骤不行是吧，那就别怪我了。直接从加密过程入手，判断出 flag 的字符 ascii 编码应该不会超出正常编码取值范围（33-125），于是直接暴力枚举，不断将正常字符进行加密，然后与加密后的 flag 进行比对，最终暴力枚举出了真正的 flag。（好在文件里的公钥私钥及一系列参数都给出了，且最终的 flag 字符比较特殊，数据经过加密后没有重复或异常，不然真就部好说了。。。）

```python

m = 'ƥĭ₅ŬSOHĮSIWÎ`ɿ`W?5ɣ?`Ė?`ɣoɿɿMWŬSHY5₅?MWĖ₅SHYoŬŬ5ÉĖ?`WŔÉSHY5ŬÀɿACKŭ'

flag = ''


for i in m:
    p = ord(i)
    for j in range(33,127):
        if (168**11*j)%487==p:
            # print(chr(j))
            flag+=chr(j)

print(flag)

# chr: 33 - 126

# VYctf{ElG4m4l_15_4n_45ymmetr1c_encrypt10n_4lg0r1thm}
```

运行 3

```
D:\app_2\python-3.11.4\python.exe D:\Document\pythonProject\vyctf\elgamal\3.py
VYctf{ElG4m4l_15_4n_45ymmetr1c_encrypt10n_4lg0r1thm}

进程已结束，退出代码为 0
```