1.Prerequisite

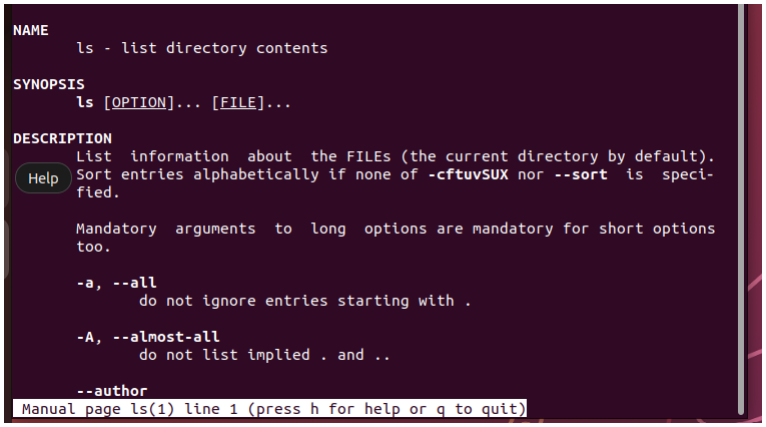
1.1 Challenge 1

Shell 命令用法介绍

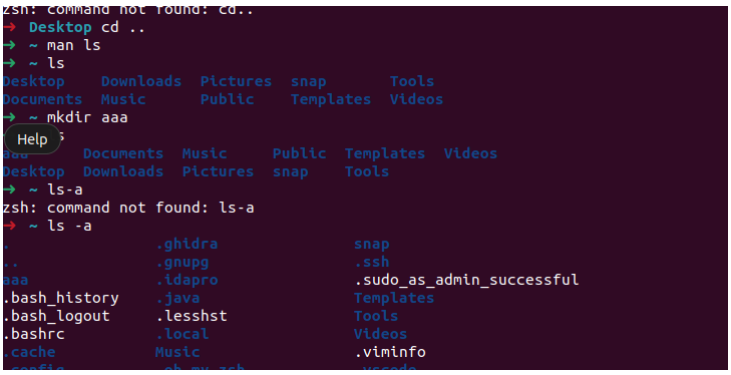
- `ls`: 列出当前路径下的文件和目录
  - `-a`: 列出所有文件和目录，包括隐藏文件
  - `-l`:列出详细信息，
- `cd path`: 用于更改当前工作路径
  - `path`中 ~ 代表 home，. 代表当前路径，.. 代表上一级路径。
- `mkdir`: 用于创建新目录
- `man`:查看`man`文档，`man` 是 "manual" 的缩写，是一个用于查看命令、函数、配置文件和系统调用的手册页的命令

Linux 环境下的实操截图

`man`



`cd ls ls -a mkdir`



1.2 Challenge 2

```
#!/usr/bin/python3

data = input("give me your string: ")
print("length of string:", len(data))

data_old = data
data_new = ""
for d in data:
    if d in 'abcdefghijklmnopqrstuvwxyz':
        data_new += chr(ord(d) - 32)
    elif d in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ':
        data_new += chr(ord(d) + 32)
    else:
        data_new += d

print("now your string:", data_new)
```

Python代码功能解释

- 程序读入用户输入的字符串长度并写入变量`data`,在屏幕显示该字符串的长度。
- 程序将用户输入的字符串中大写字母转化为小写字母，小写字母转化为大写字母，组成`data_new`，并输出。

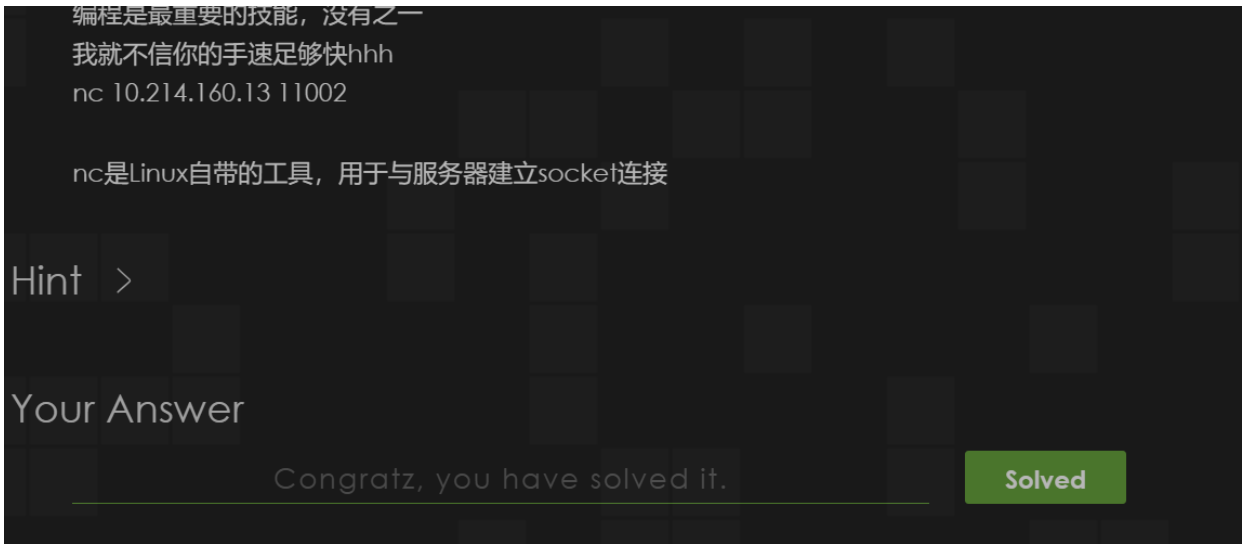
Caluclator

完整代码

```
import socket
from pwn import *
context.log_level = 'debug'

HOST = "10.214.160.13"
PORT = 11002

s = remote(HOST,PORT)

def recv_one_line(socket):
    buf = b""
    while True:
        data = socket.recv(1)
        if data == b'\n':
            return buf
        buf += data

def recv_one_question(socket):
    buf = b""
    while True:
        data = socket.recv(1)
        if data == b'=':
            return buf
```

```
        buf += data


recv_one_line(s)    # ================================================
recv_one_line(s)    # Mom: finish these 10 super simple calculations,
recv_one_line(s)    #       and you will get a flag
recv_one_line(s)    # Melody: that's easy...
recv_one_line(s)    # Mom: yep, in 10 seconds
recv_one_line(s)    # ================================================
recv_one_line(s)    #


question_1 = recv_one_question(s)
a = eval(question_1)
a_str = str(a)
a_bytes = a_str.encode()
s.send(a_bytes)
s.send(b'\n')

while True:
    recv_one_line(s)
    recv_one_line(s)

    question_2 = recv_one_question(s)
    int_answer_2 = eval(question_2)
    str_answer_2 = str(int_answer_2)
    byte_answer_2 = str_answer_2.encode()
    s.send(byte_answer_2)
    s.send(b'\n')
```

成功解决截图



正确的flag

> AAA{melody_loves_doing_calculus_qq_qun_386796080}

1.3 Challenge 3

1. What is the value of dh after line 138 executes? (Answer with a one-byte hex value)

```
xor dh, dh
```

异或运算 dh=0x00

2. What is the value of dl after line 141 executes? (Answer with a one-byte hex value)

```
and dl, 0
```

dl = 0x00

3. What is the value of di after line 161 executes? (Answer with a two-byte hex value)

```
mov dx,0xffff
not dx
mov bp,dx
mov di,bp
```

di的值即为dx的值

即为0x0000

4. What is the value of ax after line 178 executes? (Answer with a two-byte hex value)

```
mov ax ,0x0003
int 0x10
mov al,'t'
mov ah,0x0e
```

1. `mov ax, 0x0003`: 数0x0003移动到AX寄存器中。因此，AX的值为0x0003。
2. `mov al, 't'`: 将字符't'移动到AL寄存器中。因此，AL的值为ASCII码对应的数值，即0x74。
3. `mov ah, 0x0e`: 将立即数0x0e移动到AH寄存器中。因此，AH的值为0x0e。

故ax的值为0x0e74

5. What is the value of ax after line 208 executes for the third time? (Answer with a two-byte hex value)

ah值为0x0e,循环三次后，si指向.string_to_printal的第三个字节，值为0x4f，然后赋值给al

故ax的值为0x0e4f

6. What is the value of dx after line 224 executes? (Answer with a two-byte hex value)

dx的值为0x030f

flag

由上述可得flag 为ACTF{We1com3_7o_R3_00_00_0000_0e74_0e4f_030f}

2.Web

由hint中的浏览器开发工具，获得网页的源代码，利用python爬虫不断爬取新网页的cookie和token，最后得到flag

爬虫相关知识

- requests模块：用于发送HTTP请求的库，提供API使得发送GET,POST请求及处理相应很方便
- tqdm模块：用于在命令行界面显示进度条，可以方便显示迭代过程的进度使得程序运行过程更加可视化
- re模块：用于处理字符串匹配和搜索，用于从文本中提取特定模式的内容

代码及解释

```
import requests
import tqdm
import re

site = "http://pumpk1n.com/"
lab0_suffix = "lab0.php"
flag_suffix = "flag.php"
**site变量表示网站的基础URL。_suffix表示页面的路径后缀，要构建完整的URL，需要将这些部分拼接起来**

r = requests.get(site + lab0_suffix)
assert(r.status_code == 200)
cookie = r.cookies['PHPSESSID']
cookie_data = f'PHPSESSID={cookie}'
print(f"get very first cookie: {cookie_data}")
**使用requests模块发送了一个GET请求，访问了由site和lab0_suffix构建的完整URL，然后用assert语句对响应的状态码进行断言，确保状态码为200（表示成功）。之后从响应对象获取了名为‘PHPSESSID’的cookie值，并将其保存到cookie变量中，构建cookie_data字符串以便进一步使用**

def get_token_from_data(data:str):
```

```
    pattern = '/flag.php?token='
    s = data.find(pattern)
    return data[s + len(pattern) : s + len(pattern) + 16]
**这个函数get_token_from_data接收一个字符串data作为参数，并尝试从中提取出一个名为token的值。函数通过搜索指定的模式/flag.php?token=来定位data中的相关部分，然后截取出16个字符的token值并返回。
**

token_data = get_token_from_data(r.text)
print(f"get very first token :{token_data}")

for i in tqdm.tqdm(range(1337)):
    r = requests.get(site + flag_suffix + f'?token={token_data}',headers =
{'Cookie':cookie_data})
    r = requests.get(site + lab0_suffix,headers = {'Cookie':cookie_data})
    token_data = get_token_from_data(r.text)

r = requests.get(site + flag_suffix + f'?token={token_data}',headers =
{'Cookie':cookie_data})
print(r.text)
```

flag

flag{56297ad00e70449a16700a77bf24b071}

3.Pwn

3.1

Bug写在注释之中

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[])
{
    int offset;
    int size;
    char buffer[64];
    char *ptr;

    printf("please input: ");
    scanf("%s", buffer);//没检查%s的长度，可能会使buffer溢出
    printf("you input %d characters\n", strlen(buffer));
    printf("your data: %s\n", buffer);

    printf("index: ");
    scanf("%d", &offset);
    getchar();
    buffer[offset] = getchar();//没有检查offset有没有意义
```

```
    printf("size: ");
    scanf("%d", &size);
    if (size >= strlen(buffer))
        printf("size too large");
    else {
        ptr = malloc(strlen(buffer));//没有判断是否成功申请到内存
        memcpy(ptr, buffer, size);
        free(ptr);
    }

    free(ptr);//多余，否则可能会导致程序崩溃
    return 0;
}
```

3.2

- buffer在字符串较长时，程序没有直接崩溃，但是内存溢出
- 对于%s长度的问题，在输入数值比较小的时候程序不会崩溃，但当长度过大时程序崩溃
- 如果内存申请失败，程序崩溃
- 程序因double free 而崩溃

3.3

no_program.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[])
{
    int offset;
    int size;
    char buffer[64];
    char *ptr;

    printf("please input: ");
    scanf("%s", buffer);
    while(getchar()!='\n')//读取多余字符串防止溢出
    printf("you input %d characters\n", strlen(buffer));
    printf("your data: %s\n", buffer);

    printf("index: ");
    scanf("%d", &offset);
    getchar();
    buffer[offset] = getchar();

    printf("size: ");
    scanf("%d", &size);
    if (size >= strlen(buffer))
```

```
        printf("size too large");
    else {
        ptr = malloc(strlen(buffer));
        if(ptr==NULL){
            printf("Malloc failed");
            return 0;
        }
        memcpy(ptr, buffer, size);
        free(ptr);
    }

    free(ptr);
    return 0;
}
```

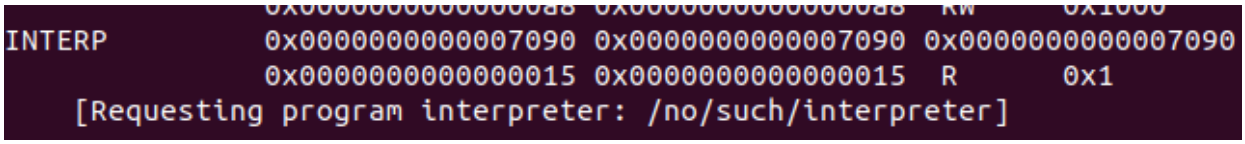
4.Reverse

1.可执行文件的入口点地址

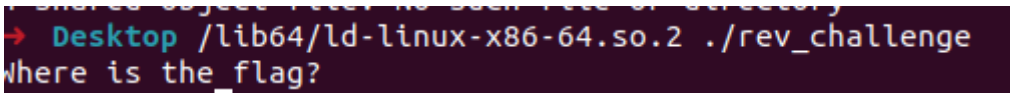
由hint中的文档可得入口点地址信息包含在ELF Header中，从而执行 `readelf -h rev_challenge` 得到入口点地址

2.可执行文件无法运行的原因，使其正常运行的方法

执行 `readelf -h rev_challenge` 发现异常



查看 ELF Header 得知其是X86-64架构，进行动态链接可得：



3.flag

通过IDA打开rev_challeng文件从main函数开始看，发现函数 `wh4t_the_h3ll_i5_this`，双击进入，注意到 `lea rax fl4g` `mov rdi,rax` 得知rdi存的是flag的地址，依次点击之后 `call` 的函数，可得到flag:AAA{hope_u_have_fun~}

5.Misc

5.1 Challenge 1

解题过程

- 使用Cyberchef的magic工具可以得到flag：AAA{wELCOm3_7o_CTf_5umMeR_c0uR5E_2023}

5.2 Challenge 2

解题过程

- 使用Cyberchef的View bit plane 工具可以得到图片隐写中的flag前半段 AAA{gr3@t_J08!*let'5*
- 使用Cyberchef将图片作为输入，在Input可得到flag的后半段P1@y_m1SC_TOG3Th3R}
- 故flag为AAA{gr3@t_J08!_let'5_P1@y_m1SC_TOG3Th3R}

6.Crypto

解题过程

AddRoundKey

- 该步骤进行将当前状态4×4的字节矩阵与当前轮密钥进行异或运算，代码实现思路就是遍历矩阵中的所有元素，依次进行异或运算
- 代码

```
def add_round_key(s, k):
    """ Add round key to the state matrix s """
    # 创建一个新的状态矩阵，初始化为零
    result = [[0] * 4 for _ in range(4)]

    # 对每个位置上的元素进行异或运算
    # TODO: You need to finish this!
    for i in range(4):
        for j in range(4):
            result[i][j] = s[i][j]^k[i][j]

    return result
```

SubBytes

- 该步骤将状态矩阵的每个字节替换为预设16×16查找表`sbox`中的不同字节。而当前状态矩阵中的字节二进制表示中高四位是`sbox`表的行索引，字节的低四位是`sbox`表的列索引。可以通过相应操作获取该元素在`sbox`中的对应替换值。
- 具体方法
  - 高字节：二进制表示中，右移操作会将数字的所有位向右移动指定的位数，并在左侧用零填充空出的位。便可通过右移运算获得高字节
  - 低字节：可以通过掩码（低4位为1，高4位为0）与原始字节进行位与运算
  - 最后注意到`sbox`是定义为一维元组，通过`row*16+col`的方法寻找索引
- 代码

```
def sub_bytes(s, sbox):
    """ Take each byte of the state matrix s and substitute it for a different byte
in a preset 16x16 lookup table(sbox)"""
    # 创建一个新的状态矩阵，初始化为零
    result = [[0] * 4 for _ in range(4)]

    # 对每个位置上的元素使用sbox进行替换
    # TODO: You need to finish this!
    for i in range(4):
        for j in range(4):
            byte = s[i][j]
            row = byte >> 4
            col = byte & 0x0F
            result[i][j] = sbox[row*16+col]
    return result
```

Flag

> AAA{AE5_aEs_a1s}