

安恒11月赛逆向Generate

原创

C站训练营学 于 2018-11-30 14:31:44 发布 164 收藏

版权声明：本文为博主原创文章，遵循 [CC 4.0 BY-NC-SA](https://creativecommons.org/licenses/by-nc-sa/4.0/) 版权协议，转载请附上原文出处链接和本声明。

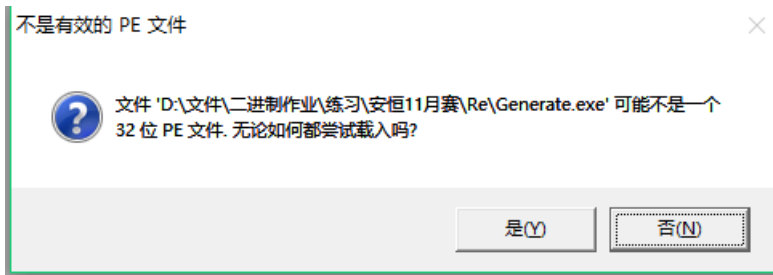
本文链接：<https://blog.csdn.net/cpongo3/article/details/89708312>

版权

一、运行程序：

□

程序不能动态调试：



载入ida:

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Readme: 读入数据0x10个
转换成为整数
输入函数, v8为输入
经过异或生成v4
v4范围应满足0x40<v4<=0x5a or v4==95 or 123 or 125
异或完数据存入byte_408040数组
输入数据右移一位
检验数组中开头字母是否为FLAG{
正确会遍历数组408040, 输出flag

程序是让用户输入一个大整数，然后该整数和程序中数组404020的每一位异或，在异或v5，v5初始为0，后面v5^=byte404020每一位，输入的整数右移一位，再进入下一次循环，满足条件则存入数组408040，

is_begin_with函数测试是否以FLAG{开头：start:内容入下图：

```
.data:0000000000404020          dd 0L0h, 0i 2h, 0L0h, 0L0h
.data:0000000000404020          db 0E1h, 61h, 0A0h, 0C0h,
.data:0000000000404040          db 5, 20h
.data:0000000000404040          public start
.data:0000000000404040          start      db 'FLAG{',0 |
.data:0000000000404046          align 20h
.data:0000000000404060          p_73208   dq offset qword_403830
.data:0000000000404068          alion 10h
```

最后等于“}”则输出flag。所以我们可以爆破，但最好的方法是Z3约束求解。

二、了解Z3模块

Z3是什么？Z3由微软开发的一套约束求解器，你可以简单的理解它是解方程的神器。在CTF题目中，我们经常遇到一些给定的条件，或是算法难以逆向、或是涉及到未知的数学技巧又或是爆破时间过长，而在我们学会使用z3后，一类问题便迎刃而解了。想了解更多关于z3知识的，这里有篇专栏：[点我](#)

Z3仅支持python2

一个简单的例子给大家介绍一下z3如何使用：

```
>>> from z3 import *
>>> x = Int('x')
>>> y = Int('y')
>>> solve(x+y==4)
[y = 0, x = 4]
```

它为我们提供了一个关于x+y==4的解，可是如果我们想要x=3呢？

```
>>> solve(x==3,x+y==4)
[y = 1, x = 3]
```

当然了，z3能做的肯定不止这么简单的运算，例如：

```
>>> from z3 import *
>>> x = Real('x')
>>> y = Real('y')
>>> solve(x**2 + y**2 == 3, x**3 == 2)
[x = 1.2599210498?, y = -1.1885280594?]
```

OK，大概了解到它是干嘛的我们就开始看一道例题吧。二进制文件可以在[这里](#)下载。

这是whctf的一道逆向题，它的核心代码如下：

```

v1 = 0;
gets(flag);
for ( i = 0; i <= 35; ++i )
{
    if ( !flag[i] )
    {
        flag[i] = 1;
        ++v1;
    }
}
if ( v1 != 9 )
    exit(0);
convert(a);
Transposition(a);
Multi(a, b);
for ( j = 0; j <= 5; ++j )
{
    for ( k = 0; k <= 5; ++k )
    {
        if ( c[0][k + 6 * j] != d[0][k + 6 * j] )
            exit(0);
    }
}
printf("congratulations!you have gottern the flag!");

```

其中convert(a)是将flag赋值给a，你可以把a当做一个6*6的矩阵。

Transposition(a)是把a的转置矩阵赋值给b

Multi(a,b)是把a和b的乘积赋值给c

而d就是堆中正确的flag经过上述运算后的结果，也就是说，如果用简单的思路去做，就是想办法爆破27位的flag添加9位1到尾部，然后经过运算结果为d中的值。但未知位数已经达到了20个，常规的爆破思路很难解决，网上的一篇writeup是经过一系列数学运算后逐行爆破，但每行依旧要消耗近10分钟的时间。在实际比赛的过程中，时间始终是最宝贵的，况且如果你对线性代数不太理解，可能会有一些棘手。

首先数学知识当然是必要的，我们应该保持着一个敬畏之心去学习这里的数学原理，但为了节省时间，或许用约束器去做会有意想不到的效果。

以下是我的脚本，注释的很详细就不多说了：

```

#coding:utf-8

...

@DateTime:      2017-11-28 10:19:29

@Version:       1.0

@author:       Unname_Bao

...

from z3 import *

```

```

import time

t1 = time.time()

#创建一个解决方案实例

solver = Solver()

#flag长度先设置为36，包括尾部的9个1

flag = [Int('flag%d%i) for i in range(36)]

#保存flag的矩阵

a = [i for i in flag]

#保存flag的转置矩阵

b = [i for i in range(36)]

#保存a*b的矩阵

c = [0 for i in range(36)]

#堆中正确flag的运算结果

d = [0x12027,0x0F296,0x0BF0E,0x0D84C,0x91D8,0x297,
      0x0F296,0x0D830,0x0A326,0x0B010,0x7627,0x230,
      0x0BF0E,0x0A326,0x8FEB,0x879D,0x70C3,0x1BD,
      0x0D84C,0x0B010,0x879D,0x0B00D,0x6E4F,0x1F7,
      0x91D8,0x7627,0x70C3,0x6E4F,0x9BDC,0x15C,
      0x297,0x230,0x1BD,0x1F7,0x15C,0x6]

#获得a的转置矩阵

for i in range(6):
    for j in range(6):
        b[i+6*j] = a[6*i+j]

#运算a*b

for i in range(6):
    for j in range(6):
        for k in range(6):
            c[j+6*i] = c[j+6*i] + a[6*i+k]*b[6*k+j]

#添加约束，正确flag的运算结果

solver.add(simplify(c[j+6*i]) == d[j+6*i])

```

```
#添加约束，除了尾部，flag的字符一定在可见字符范围内
```

```
for i in range(6,36-10):
```

```
    solver.add(flag[i]>=32)
```

```
    solver.add(flag[i]<=127)
```

```
#添加约束，由于flag有格式，前6位一定为whctf{
```

```
for i in range(6):
```

```
    solver.add(flag[i] == ord('whctf{'[i]))
```

```
#添加约束，flag的尾部为9个1
```

```
for i in range(36-9,36):
```

```
    solver.add(flag[i] == 0x1)
```

```
#添加约束，flag的最后一个肯定是}
```

```
solver.add(flag[-10] == ord('}'))
```

```
#这里一定要有，不check的话会报错
```

```
if solver.check() == sat:
```

```
    m = solver.model()
```

```
    s = []
```

```
    #获得结果
```

```
    for i in range(36):
```

```
        s.append(m[flag[i]].as_long())
```

```
    #输出flag
```

```
    print(bytes(s))
```

```
else:
```

```
    print('error')
```

```
t2 = time.time()
```

```
print(t2-t1)
```

这是最终的运行结果:

```
D:\2017_WEB_Test\ulb_manager\backend\spider>python z3test.py
```

```
b'whctf{Y0u_ar3_g00d_a7_m4th}\x01\x01\x01\x01\x01\x01\x01\x01'
```

```
4.042840003967285
```

是的，仅仅用了4s就跑出了最终结果、可见z3约束器的强大！

更多信息请看z3的官方GitHub: [点我](#)

三、题解脚本

```
from z3 import *
```

```
def gen():
    print('[*]Computing key...')
    a = [0xA4, 0x19, 0x04, 0x82, 0x7E, 0x85, 0x50, 0xA8, 0xD1, 0xEA,
         0xE3, 0xF9, 0xE8, 0xE1, 0x60, 0x3A, 0x1A, 0x0A, 0x87, 0xDD,
         0xE1, 0x61, 0xA0, 0xC0, 0x60, 0xA4, 0x48, 0x28, 0x16, 0xB,
         0x05, 0x20]
    num = BitVec('x', 64)#num范围不会超过64位

    s = Solver()#定义运算器
    s.add(num >= 2 ** 31)#添加约束条件
    s.add(num < 2 ** 32)
    num2 = 0
    b = 0
    for i in range(32):
        if i < 5:
            s.add((a[num2] ^ (num & 0xff) ^ b) & 0xff == ord("FLAG"[i]))#约束条件前几位为固定数
        elif 5 <= i < 31:
            s.add(Or(
                #Or
                And((a[num2] ^ (num & 0xff) ^ b) & 0xff <= ord("Z"),#范围在“@”“Z”之间 And
                    (a[num2] ^ (num & 0xff) ^ b) & 0xff >= ord("A")),
                (a[num2] ^ (num & 0xff) ^ b) & 0xff == ord("_"))#异或之后为“_”
            )
        else:
            s.add((a[num2] ^ (num & 0xff) ^ b) & 0xff == ord("{}"))

        b ^= a[num2]
        b &= 0xff
        num2 += 1
        num >>= 1
    if s.check() == sat:#检测是否有解 有解为sat
        a = s.model() #将结果赋给a
        print(a)
        return
    print('[*]No Result ,end')

gen()
```

得到结果:

```
C:\Users\yrl>D:\文件\二进制作业\练习\安恒11月赛\Re\Generate.exe
try to enter a number to generate the flag
3658134498
Congratulations!
flag is:
FLAG {__ZZLOZEEZ_Z__AAPHTZIZ__}
C:\Users\yrl>
```

赏

你的支持是我最大的动力!

支付宝

微信