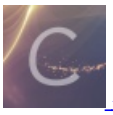
DASCTF X SU-2022-Crypto-FlowerCipher之暴力暴力求解法(z3约束器)

原创

Mango|Feng 已于 2022-03-30 21:01:48 修改 339 收藏

分类专栏： 密码学 笔记 python 文章标签： python 安全 算法

于 2022-03-27 17:32:06 首次发布



密码学 同时被 3 个专栏收录

12 篇文章 0 订阅

订阅专栏

笔记

12 篇文章 0 订阅

订阅专栏

python

10 篇文章 0 订阅

订阅专栏

题目

```
from pickle import LONG1
from secret import flag
import random

# flag = b'flag{%s}' % md5(something).hexdigest()
# note that md5 only have characters 'abcdef' and digits

def Flower(x, key):
    flower = random.randint(0, 4096)
    return x * (key ** 3 + flower)

flag = flag[5:-1]
rounds = len(flag)

L, R = 1, 0
for i in range(rounds):
    L, R = R + Flower(L, flag[i]), L

print(L, R)
'''
L=157201972689453483884294293513030069253873889272923047175945112593901941008508898527476533871972053924310530690436323403742526295294197768744108179277709223108086325816661818 99
R=139721425176294317602347104909475448503147767726747922243703132013053043430193232376860554749633894589164137720010858254771905261753520854314908256431590570426632742469003
'''
```

分析一下

```
# flag = b'flag{%s}' % md5(something).hexdigest()
# note that md5 only have characters 'abcdef' and digits
```

> 告诉了flag中含的内容是一个md5值，且md5中只含有abcdef0123456789这些字符
> (当时想都没想就把长度当成了32位了，还好题目做出来了，如果是十六位的话，可能当场暴毙)

```
def Flower(x, key):
    flower = random.randint(0, 4096)
    return x * (key ** 3 + flower)
```

> 生成一个在0-4096间的随机数flower，返回x * (key ** 3 + flower)

```
flag = flag[5:-1]
rounds = len(flag)
```

> 取了flag里面md5的值作为flag
> 将md5的长度作为循环次数

```
L, R = 1, 0
for i in range(rounds):
    L, R = R + Flower(L, flag[i]), L

print(L, R)
L=157201972689453483884294293513030069253873889272923047175945112593901941008508898527476533871972053924310530690436323403742526295294197768744108179277709223108086325816661818 99
R=139721425176294317602347104909475448503147767726747922243703132013053043430193232376860554749633894589164137720010858254771905261753520854314908256431590570426632742469003
## L比R长几位
```

> 给了L,R的初值以及经过了rounds次循环之后的L和R.

感觉这些条件可以把中间过程都推出来.

由于前几天才做了一道 LFSR 的题D3bug,用的是 z3-solve 求解,先列一点条件出来看看

条件

$$L0,R0= 1,0$$

$$L32= 157201972689453483884294293513030069253873889272923047175945112593901941008508898527476533$$

$$R32= 139721425176294317602347104909475448503147767726747922243703132013053043430193232376860554$$

$$L = R \quad + r \quad *$$

$$random \in [0,4096]$$

$$flag \in ascii\{abcdef0123456789\}$$

- 因为这里abcdef与0123456789之间还隔着一些ascii字符，用一般的逻辑式怕z3识别不了，就利用了简单粗暴的逻辑表达比如$flag<$ _ ...

当时就写了那么多条件,然后发现z3一直跑呀跑呀跑呀跑呀，没有任何回应.

然后后面就考虑到要加条件上去，又开始看代码：

```
L, R = 1, 0
for i in range(rounds):
    L, R = R + Flower(L, flag[i]), L
```

我们当时用了

$R = L \quad$ 的条件，要想逆推出$R \quad$ 来，继续分析一下

且$L \quad *$

即我们可以用$(1)\%(2)$

即$L \ \%R$ 就可以得到R

加上条件

- $L\ \%R = R$

然后来编写一个超级超级超级长的z3求解代码

Solve

```
from z3 import *
L0=Int('L0')
L1=Int('L1')
L2=Int('L2')
L3=Int('L3')
L4=Int('L4')
L5=Int('L5')
L6=Int('L6')
L7=Int('L7')
L8=Int('L8')
L9=Int('L9')
L10=Int('L10')
L11=Int('L11')
L12=Int('L12')
L13=Int('L13')
L14=Int('L14')
L15=Int('L15')
L16=Int('L16')
L17=Int('L17')
L18=Int('L18')
L19=Int('L19')
L20=Int('L20')
L21=Int('L21')
L22=Int('L22')
L23=Int('L23')
L24=Int('L24')
L25=Int('L25')
L26=Int('L26')
L27=Int('L27')
L28=Int('L28')
L29=Int('L29')
L30=Int('L30')
L31=Int('L31')
L32=Int('L32')
R0=Int('R0')
R1=Int('R1')
R2=Int('R2')
R3=Int('R3')
R4=Int('R4')
R5=Int('R5')
R6=Int('R6')
R7=Int('R7')
R8=Int('R8')
R9=Int('R9')
R10=Int('R10')
R11=Int('R11')
R12=Int('R12')
R13=Int('R13')
R14=Int('R14')
R15=Int('R15')
R16=Int('R16')
R17=Int('R17')
R18=Int('R18')
```

```
R19=Int('R19')
R20=Int('R20')
R21=Int('R21')
R22=Int('R22')
R23=Int('R23')
R24=Int('R24')
R25=Int('R25')
R26=Int('R26')
R27=Int('R27')
R28=Int('R28')
R29=Int('R29')
R30=Int('R30')
R31=Int('R31')
R32=Int('R32')
flag0=Int('flag0')
flag1=Int('flag1')
flag2=Int('flag2')
flag3=Int('flag3')
flag4=Int('flag4')
flag5=Int('flag5')
flag6=Int('flag6')
flag7=Int('flag7')
flag8=Int('flag8')
flag9=Int('flag9')
flag10=Int('flag10')
flag11=Int('flag11')
flag12=Int('flag12')
flag13=Int('flag13')
flag14=Int('flag14')
flag15=Int('flag15')
flag16=Int('flag16')
flag17=Int('flag17')
flag18=Int('flag18')
flag19=Int('flag19')
flag20=Int('flag20')
flag21=Int('flag21')
flag22=Int('flag22')
flag23=Int('flag23')
flag24=Int('flag24')
flag25=Int('flag25')
flag26=Int('flag26')
flag27=Int('flag27')
flag28=Int('flag28')
flag29=Int('flag29')
flag30=Int('flag30')
flag31=Int('flag31')
random0=Int('random0')
random1=Int('random1')
random2=Int('random2')
random3=Int('random3')
random4=Int('random4')
random5=Int('random5')
random6=Int('random6')
random7=Int('random7')
random8=Int('random8')
random9=Int('random9')
random10=Int('random10')
random11=Int('random11')
random12=Int('random12')
```

```
random13=Int('random13')
random14=Int('random14')
random15=Int('random15')
random16=Int('random16')
random17=Int('random17')
random18=Int('random18')
random19=Int('random19')
random20=Int('random20')
random21=Int('random21')
random22=Int('random22')
random23=Int('random23')
random24=Int('random24')
random25=Int('random25')
random26=Int('random26')
random27=Int('random27')
random28=Int('random28')
random29=Int('random29')
random30=Int('random30')
random31=Int('random31')
so=Solver()
so.add(L1==R0+(L0*(flag0**3+random0)))
so.add(L2==R1+(L1*(flag1**3+random1)))
so.add(L3==R2+(L2*(flag2**3+random2)))
so.add(L4==R3+(L3*(flag3**3+random3)))
so.add(L5==R4+(L4*(flag4**3+random4)))
so.add(L6==R5+(L5*(flag5**3+random5)))
so.add(L7==R6+(L6*(flag6**3+random6)))
so.add(L8==R7+(L7*(flag7**3+random7)))
so.add(L9==R8+(L8*(flag8**3+random8)))
so.add(L10==R9+(L9*(flag9**3+random9)))
so.add(L11==R10+(L10*(flag10**3+random10)))
so.add(L12==R11+(L11*(flag11**3+random11)))
so.add(L13==R12+(L12*(flag12**3+random12)))
so.add(L14==R13+(L13*(flag13**3+random13)))
so.add(L15==R14+(L14*(flag14**3+random14)))
so.add(L16==R15+(L15*(flag15**3+random15)))
so.add(L17==R16+(L16*(flag16**3+random16)))
so.add(L18==R17+(L17*(flag17**3+random17)))
so.add(L19==R18+(L18*(flag18**3+random18)))
so.add(L20==R19+(L19*(flag19**3+random19)))
so.add(L21==R20+(L20*(flag20**3+random20)))
so.add(L22==R21+(L21*(flag21**3+random21)))
so.add(L23==R22+(L22*(flag22**3+random22)))
so.add(L24==R23+(L23*(flag23**3+random23)))
so.add(L25==R24+(L24*(flag24**3+random24)))
so.add(L26==R25+(L25*(flag25**3+random25)))
so.add(L27==R26+(L26*(flag26**3+random26)))
so.add(L28==R27+(L27*(flag27**3+random27)))
so.add(L29==R28+(L28*(flag28**3+random28)))
so.add(L30==R29+(L29*(flag29**3+random29)))
so.add(L31==R30+(L30*(flag30**3+random30)))
so.add(L32==R31+(L31*(flag31**3+random31)))
so.add(R1==L0)
so.add(R2==L1)
so.add(R3==L2)
so.add(R4==L3)
so.add(R5==L4)
so.add(R6==L5)
so.add(R7==L6)
so.add(R8==L7)
```

```
so.add(R9==L8)
so.add(R10==L9)
so.add(R11==L10)
so.add(R12==L11)
so.add(R13==L12)
so.add(R14==L13)
so.add(R15==L14)
so.add(R16==L15)
so.add(R17==L16)
so.add(R18==L17)
so.add(R19==L18)
so.add(R20==L19)
so.add(R21==L20)
so.add(R22==L21)
so.add(R23==L22)
so.add(R24==L23)
so.add(R25==L24)
so.add(R26==L25)
so.add(R27==L26)
so.add(R28==L27)
so.add(R29==L28)
so.add(R30==L29)
so.add(R31==L30)
so.add(R32==L31)
so.add(R0==0)
so.add(L0==1)
so.add(R0==L1%R1)
so.add(R1==L2%R2)
so.add(R2==L3%R3)
so.add(R3==L4%R4)
so.add(R4==L5%R5)
so.add(R5==L6%R6)
so.add(R6==L7%R7)
so.add(R7==L8%R8)
so.add(R8==L9%R9)
so.add(R9==L10%R10)
so.add(R10==L11%R11)
so.add(R11==L12%R12)
so.add(R12==L13%R13)
so.add(R13==L14%R14)
so.add(R14==L15%R15)
so.add(R15==L16%R16)
so.add(R16==L17%R17)
so.add(R17==L18%R18)
so.add(R18==L19%R19)
so.add(R19==L20%R20)
so.add(R20==L21%R21)
so.add(R21==L22%R22)
so.add(R22==L23%R23)
so.add(R23==L24%R24)
so.add(R24==L25%R25)
so.add(R25==L26%R26)
so.add(R26==L27%R27)
so.add(R27==L28%R28)
so.add(R28==L29%R29)
so.add(R29==L30%R30)
so.add(R30==L31%R31)
so.add(R31==L32%R32)
so.add(L32==157201972689453483884294293513030069253873889272923047175945112593901941008508898527476533871972053922
```

```
43105306904363234037425262952941977687441081792777092231080863258166618189 9)
so.add(R32==139721425176294317602347104909475448503147767726747922243703132013053043430193232376860554749633894589164137720010858254771905261753520854314908256431590570426632742469003)
so.add(random0<=4096)
so.add(random0>=0)
so.add(random1<=4096)
so.add(random1>=0)
so.add(random2<=4096)
so.add(random2>=0)
so.add(random3<=4096)
so.add(random3>=0)
so.add(random4<=4096)
so.add(random4>=0)
so.add(random5<=4096)
so.add(random5>=0)
so.add(random6<=4096)
so.add(random6>=0)
so.add(random7<=4096)
so.add(random7>=0)
so.add(random8<=4096)
so.add(random8>=0)
so.add(random9<=4096)
so.add(random9>=0)
so.add(random10<=4096)
so.add(random10>=0)
so.add(random11<=4096)
so.add(random11>=0)
so.add(random12<=4096)
so.add(random12>=0)
so.add(random13<=4096)
so.add(random13>=0)
so.add(random14<=4096)
so.add(random14>=0)
so.add(random15<=4096)
so.add(random15>=0)
so.add(random16<=4096)
so.add(random16>=0)
so.add(random17<=4096)
so.add(random17>=0)
so.add(random18<=4096)
so.add(random18>=0)
so.add(random19<=4096)
so.add(random19>=0)
so.add(random20<=4096)
so.add(random20>=0)
so.add(random21<=4096)
so.add(random21>=0)
so.add(random22<=4096)
so.add(random22>=0)
so.add(random23<=4096)
so.add(random23>=0)
so.add(random24<=4096)
so.add(random24>=0)
so.add(random25<=4096)
so.add(random25>=0)
so.add(random26<=4096)
so.add(random26>=0)
so.add(random27<=4096)
so.add(random27>=0)
so.add(random28<=4096)
```

```
so.add(random28>=0)
so.add(random29<=4096)
so.add(random29>=0)
so.add(random30<=4096)
so.add(random30>=0)
so.add(random31<=4096)
so.add(random31>=0)
so.add(flag0>=48)
so.add(flag0<=102)
so.add(flag0!=58)
so.add(flag0!=59)
so.add(flag0!=60)
so.add(flag0!=61)
so.add(flag0!=62)
so.add(flag0!=63)
so.add(flag0!=64)
so.add(flag0!=65)
so.add(flag0!=66)
so.add(flag0!=67)
so.add(flag0!=68)
so.add(flag0!=69)
so.add(flag0!=70)
so.add(flag0!=71)
so.add(flag0!=72)
so.add(flag0!=73)
so.add(flag0!=74)
so.add(flag0!=75)
so.add(flag0!=76)
so.add(flag0!=77)
so.add(flag0!=78)
so.add(flag0!=79)
so.add(flag0!=80)
so.add(flag0!=81)
so.add(flag0!=82)
so.add(flag0!=83)
so.add(flag0!=84)
so.add(flag0!=85)
so.add(flag0!=86)
so.add(flag0!=87)
so.add(flag0!=88)
so.add(flag0!=89)
so.add(flag0!=90)
so.add(flag0!=91)
so.add(flag0!=92)
so.add(flag0!=93)
so.add(flag0!=94)
so.add(flag0!=95)
so.add(flag0!=96)
so.add(flag1>=48)
so.add(flag1<=102)
so.add(flag1!=58)
so.add(flag1!=59)
so.add(flag1!=60)
so.add(flag1!=61)
so.add(flag1!=62)
so.add(flag1!=63)
so.add(flag1!=64)
so.add(flag1!=65)
so.add(flag1!=66)
```

```
so.add(flag1!=67)
so.add(flag1!=68)
so.add(flag1!=69)
so.add(flag1!=70)
so.add(flag1!=71)
so.add(flag1!=72)
so.add(flag1!=73)
so.add(flag1!=74)
so.add(flag1!=75)
so.add(flag1!=76)
so.add(flag1!=77)
so.add(flag1!=78)
so.add(flag1!=79)
so.add(flag1!=80)
so.add(flag1!=81)
so.add(flag1!=82)
so.add(flag1!=83)
so.add(flag1!=84)
so.add(flag1!=85)
so.add(flag1!=86)
so.add(flag1!=87)
so.add(flag1!=88)
so.add(flag1!=89)
so.add(flag1!=90)
so.add(flag1!=91)
so.add(flag1!=92)
so.add(flag1!=93)
so.add(flag1!=94)
so.add(flag1!=95)
so.add(flag1!=96)
#这里只展示flag1的条件
#flag2-flag31类似 代码量太大了 容易卡
print(so.check())
print(so.model())
```

代码有很多语句是冗余的但是思路是很清晰的!容易让人理解

```
output:
sat
[flag4 = 55,
 random16 = 4070,
 random17 = 1206,
 random30 = 510,
 random21 = 2623,
 flag26 = 49,
 random13 = 3129,
 random12 = 3927,
 flag16 = 54,
 random0 = 666,
 random10 = 2967,
 random5 = 3868,
 random7 = 330,
 flag20 = 100,
 flag5 = 98,
 random3 = 217,
 flag6 = 54,
 random18 = 536,
 flag27 = 48,
 flag28 = 56,
```

flag17 = 55,
flag10 = 50,
flag12 = 100,
random8 = 259,
random27 = 3540,
random24 = 1583,
random1 = 1021,
flag19 = 100,
flag7 = 54,
flag9 = 102,
flag31 = 48,
random28 = 2804,
flag13 = 51,
flag11 = 54,
flag1 = 101,
random11 = 2623,
flag29 = 50,
random22 = 677,
flag21 = 99,
flag23 = 98,
flag14 = 56,
flag0 = 51,
random15 = 3699,
random26 = 2014,
random9 = 427,
random4 = 398,
random19 = 899,
random2 = 1240,
random20 = 2182,
random29 = 1341,
flag30 = 53,
flag24 = 57,
flag8 = 101,
flag18 = 49,
flag22 = 98,
random25 = 2910,
flag15 = 101,
random31 = 1919,
random6 = 2545,
flag25 = 99,
random14 = 2360,
flag2 = 56,
random23 = 1553,
flag3 = 48,
L1 = 133317,
L2 = 137492755075,
L3 = 24316418691677517,
L4 = 2694478038943586736328,
L5 = 449366186013055209469307061,
L6 = 42467800775619243430000691780 4988,
L7 = 67952303343509961405922862120527631953,
L8 = 10722465754210488857842384539746544074196670,
L9 = 11050144307727113700681557772687121323224647867153,
L10 = 11731219952144596819377276074864534430521345582519171825,
L11 = 15012090236271377654929790011728714352432121514814555087969 28,
L12 = 24032404897712882341661912618013874552864463812473311361929298 4561,
L13 = 24126780151896321732980332725414112938350849705389215270795740362016 7975,
L14 = 32759342090485149698017824597983901673872922475506121132811189377165700630061,
L15 = 58303766681374528041733835679805862115633483798841859117870963932984001389559045 11,

L16 = 60286094748868855416057637589899439673544861264741211552633637918033569330575709650040 61,
L17 = 97382540292220854574588289584885499239062014816543403507419639265695055521782006839992 1894085,
L18 = 16319463485313523977952768711085273223880245901706608715824302683310779478576086181558 4881897662446,
L19 = 19287157921091613716265688246942013055491723611322575658962386161345041119412098008892 719335475158074595,
L20 = 19304497076225869711849746340455541612339463403087957113496859433662333338211557279788 474751973335123601723351,
L21 = 19346619488865481717482094100686681292384530125288986759529832156605546935716879938892 385301891033660176897469426477,
L22 = 18822751726365286700612339826340137082689797360168751039458371318582478795225200597245 268849966216725478600774872948579145,
L23 = 17728566345779292838907909381612640668036643431117165902908905722221490552536570008262 521006387722966311695266888986102760148482,
L24 = 16713517279670522179142602316669021266414545548551242366498025076135157482269671171234 675566764239156725485371108735804221489129242235,
L25 = 31216839034454700168773179831370810254374558000442434876761522975231290796306215932310 643336662200537429469786405169338361618397061078328 42,
R32 = 13972142517629431760234710490947544850314776772674792224370313201305304343019323237686 055474963389458916413772001085825477190526175352085431490825643159057042663274246900 3,
L32 = 15720197268945348388429429351303006925387388927292304717594511259390194100850889852747 653387197205392431053069043632340374252629529419776874410817927770922310808632581666181899,
R31 = 93529842067175423083301473884973043258816923803322817346958313147641908479469551176114 627830960677002749066727161079662426939203458617508839623564153775609373618536 6,
R30 = 74029683208955321169307683700989297646780650936025167511852256099680539613981956717966680 350673894083067361794621735938827959163846598026491898008516652191983 61,
R29 = 41491807647864532203061547188977816042392604608090542687445179257686072390683442091157724 792609311622180322599523073162631870961894947012137520634996058265,
R28 = 36354228126052712064150782639437657942700212489125672681170492545245593389230677757003602 867732302125526688020601749936336335674361336915566850355706 1,
R27 = 30380508700049759469348282792299980900016299429716727059463717436866849535343727676090805602742030278498839252924843300328659636627629870215722 13,
L0 = 1,
R0 = 0,
L31 = 13972142517629431760234710490947544850314776772674792224370313201305304343019323237686 055474963389458916413772001085825477190526175352085431490825643159057042663274246900 3,
L30 = 93529842067175423083301473884973043258816923803322817346958313147641908479469551176114 627830960677002749066727161079662426939203458617508839623564153775609373618536 6,
L29 = 74029683208955321169307683700989297646780650936025167511852256099680539613981956717966680 350673894083067361794621735938827959163846598026491898008516652191983 61,
L28 = 41491807647864532203061547188977816042392604608090542687445179257686072390683442091157724 792609311622180322599523073162631870961894947012137520634996058265,
L27 = 36354228126052712064150782639437657942700212489125672681170492545245593389230677757003602 867732302125526688020601749936336335674361336915566850355706 1,
L26 = 30380508700049759469348282792299980900016299429716727059463717436866849535343727676090805602742030278498839252924843300328659636627629870215722 13,
R26 = 31216839034454700168773179831370810254374558000442434876761522975231290796306215932310 643336662200537429469786405169338361618397061078328 42,
R25 = 16713517279670522179142602316669021266414545548551242366498025076135157482269671171234 675566764239156725485371108735804221489129242235,
R24 = 17728566345779292838907909381612640668036643431117165902908905722221490552536570008262 521006387722966311695266888986102760148482,
R23 = 18822751726365286700612339826340137082689797360168751039458371318582478795225200597245 268849966216725478600774872948579145,
R22 = 19346619488865481717482094100686681292384530125288986759529832156605546935716879938892 385301891033660176897469426477,
R21 = 19304497076225869711849746340455541612339463403087957113496859433662333338211557279788 474751973335123601723351,
R20 = 19287157921091613716265688246942013055491723611322575658962386161345041119412098008892 719335475158074595,
R19 = 16319463485313523977952768711085273223880245901706608715824302683310779478576086181558 4881897662446,
R18 = 97382540292220854574588289584885499239062014816543403507419639265695055521782006839992 1894085,
R17 = 60286094748868855416057637589899439673544861264741211552633637918033569330575709650040 61,

```
R17 = 6028609474886885541605763758989943967354486126474121155263363791803356933057570965004061,
R16 = 583037666813745280417338356798058621156334837988418591178709639329840013895590451 1,
R15 = 32759342090485149698017824597983901673872922475506121132811189377165700630061,
R14 = 2412678015189632173298033272541411293835084970538921527079574036201 67975,
R13 = 240324048977128823416619126180138745528644638124733113619292984561,
R12 = 150120902362713776549297900117287143524321215148145550879692 8,
R11 = 117312199521445968193772760748645344305213455825191718 25,
R10 = 110501443077271137006815577726871213232246478671 53,
R9 = 10722465754210488857842384539746544074196670,
R8 = 6795230334350996140592286212052 7631953,
R7 = 4246780077561924343000069178 04988,
R6 = 449366186013055209469307061,
R5 = 26944780389435867363 28,
R4 = 24316418691677517,
R3 = 137492755075,
R2 = 133317,
...]
```

我们整理一下 flag 有关的值

```
flag=[51,101,56,48,55,98,54,54,101,102,50,54,100,51,56,101,54,55,49,100,100,99,98,98,57,99,49,48,56,50,53,48]
print("flag{",end="")
for i in flag:
    print(chr(i),end="")
print("}")
#flag{3e807b66ef26d38e671ddcbb9c108250}
```

总结一下

z3 是真的好用把条件都加进去就可以把整个过程都求解出来了。

代码不是一行一行打的，一千多行的代码手打会很枯燥而且很花时间！
后面重新写了一篇DASCTF X SU-2022-Crypto-FlowerCipher(利用已知条件爆破)

参考

求余条件是看到这个wp才加上去的:

https://blog.csdn.net/qq_42880719/article/details/123763744