hitcon2016 misc writeup

转载

skd12344321 于 2016-11-05 20:01:51 发布 279 收藏
分类专栏： misc 文章标签： hack


misc 专栏收录该内容
1 篇文章 0 订阅
订阅专栏

Beelzemon

这是一道ppc题目

```
Beelzemon gives you two integers
1 <= k <= n <=
20.



It wants to know
if you can
split a
set {a |
-(2**n) <= a <= (
2**n) -
1} into two sets A, B s.t. |
A|
= |B|
and
sum({a**k |
a
in A}) =
sum({b**k |
b
in B}).



Give Beelzemon either A or B to save your life. (separate the numbers
by space)
```

简单描述下题意

大概有几点：

1、k\n在1到20之间，并且k<=n
2、a是一个从　　到　　的整数集合
3、A\B中元素数量相等，并且和相等

这时候我们我们需要一些理论支持了，当天在做题的时候，我找到了这样一篇文章

https://zhuanlan.zhihu.com/p/20559045

这里有一个理论

对于数字 1 到 2^n（$n \in N, n \geq 2$），如果我们按照以下方式分组：

- **1 在 A 组，2 在 B 组；**
- **假设 1 到 2^k（$k \in N, k \geq 1$）已经分组完毕，其中 A 组从小到大依次是 $a_1, a_2, \ldots, a_{2^{k-1}}$，B 组从小到大依次是 $b_1, b_2, \ldots, b_{2^{k-1}}$，那么，将 2^k+1 到 2^{k+1} 按照如下方式分组：$a_{2^{k-1}+t} = b_t + 2^k$，$b_{2^{k-1}+t} = a_t + 2^k$（$t = 1, 2, \ldots, 2^{k-1}$）。**

那么，对于最终的分组：$A: a_1, a_2, \ldots, a_{2^{n-1}}$，$B: b_1, b_2, \ldots, b_{2^{n-1}}$，它们的 p 次幂和（$p = 1, 2, \ldots, n-1$）都相等。

所以

假设 n = k 时，$A: a_1, a_2, \ldots, a_{2^{k-1}}$，$B: b_1, b_2, \ldots, b_{2^{k-1}}$，两组数的 1,2,...k-1 次幂和都相等。

那么当 n = k + 1 时，

$$A: a_1, a_2, \ldots, a_{2^{k-1}}, b_1 + 2^k, b_2 + 2^k, \ldots, b_{2^{k-1}} + 2^k$$
$$B: b_1, b_2, \ldots, b_{2^{k-1}}, a_1 + 2^k, a_2 + 2^k, \ldots, a_{2^{k-1}} + 2^k$$

但是我们又遇到了一个问题，题目中需要对包括负数的集合做处理（当时也没想明白），后看来看了wp才明白这里

https://github.com/JulesDT/ctfWriteUps/tree/master/Hitcon%20Quals%202016/Beelzemon%20-%20PPC%20-%20150%20pts

贴上解题脚本

```
import socket

import re

import operator

import time


def find_partition(int_list,n,k):

    len_A=
0; len_B=
```

```
0; len_B=
0; sum_A=
0; sum_B=
0


    Aret =
""; Bret =
""



for i
in range(
0,len(int_list)):



        int_list[i] +=
2**n



    int_list=int_list[::
-1]


for nb
in int_list:


if nb ==
0:


if len_A < len_B:


                len_A+=
1


                Aret+= str(
-2**n)+
" "


```

```
        else:
18
                        len_B+=
        1
19
                        Bret+= str(
        -2**n)+
20      " "
21      else:
22      if sum_A < sum_B:
23                      sum_A+=(nb**k)
24                      len_A+=
        1
25
                        Aret+=str(nb-(
        2**n))+
        " "
26
        else:
27
                        sum_B+=(nb**k)
28
                        len_B+=
        1
29
                        Bret+=str(nb-(
30      2**n))+
        " "
```

```
31
        return (Aret)
32
        def main():
33
            begin = time.time()
34
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
35
            s.connect((
        '52.198.217.117',
36      6666))
        while
37      True:
38
                data = s.recv(
        2048)
39
        print
        "Received:", data
40
        if len(repr(data)) <=
        2 :
41
        break;
42
                mgex = re.search(
43      '([0-9]+) ([0-9]+)', repr(data))
```

```
44          if mgex !=
            None:
45
                        n = long(mgex.group(
            1));
46
                        k = long(mgex.group(
            2));
47
                        mySet = range(
            -2**n,
48          2**n);
49                      partition = find_partition(mySet,n,k)
50                      s.send(partition+
            '\n');
51
            print
            "Connection closed."
52
                s.close()
53
            print
            "Process duration :", time.time() - begin
            main()
```

这里我们看到列表通过了处理

```
int_list
[i] +=
2**n
```

通过这样的处理，所有本来的负数就被处理成了正数，然后再插入结果列表的时候在去掉，然后再利用刚才的理论，就可以得到结果了

hackpad

刚开始看到上来get 3次，然后就post暴力跑什么东西，错误的返回500，正确的返回200，那么看上去像是在跑cbc的iv了。

所以这里是padding oracle attack

web小白上也有提到这种攻击方式

http://blog.zhaojie.me/2010/10/padding-oracle-attack-in-detail.html

简单来说就是这个逻辑

```
接受到正确的密文之后（填充正确且包含合法的值），应用程序正常返回（
200 - OK）。
```

```
接受到非法的密文之后（解密后发现填充不正确），应用程序抛出一个解密异常（
500 - Internal Server Error）。
```

```
接受到合法的密文（填充正确）但解密后得到一个非法的值，应用程序显示自定义错误消息（
200 - OK）。
```

那么我们需要把每一次跑到的xor 0x01,0x02,0x03，然后异或对应密文。

```
def xor_c(s, key):
```

```
 r =
''
```

```
for c
in s:
```

```
        r += chr(key ^ ord(c))
```

```
4
5       return r
6
7       def xor_ss(s1, s2):
8        r =
        ''
9       for i
        in range(len(s1)):
10            r += chr(ord(s1[i]) ^ ord(s2[i]))
11
12      return r
13
14      def main():
15       bs = open(
        'hackpad.pcap',
        'rb').read()
16
17       i =
        0x2634
        -1
```

```
 cs = []

 j = 0

 flag = ''

while i!= -1:

      i = bs.find('POST / HTTP/1.1', i+1)

      msg = bs.find('msg=', i)

      res = bs.find('HTTP/1.1', msg)

if bs[res+9] == '2':

           iv = bs[msg+4:msg+36]

           c = bs[msg+36:msg+
```

```
68]


31
if iv[
0] !=
'0':

32


#print iv,

33


                iv = xor_c(iv.decode(
'hex'),
34      0x10)


35                 cs.append(c.decode(
'hex'))


36
if j !=
0:

37


                    flag += xor_ss(cs[j
-1], iv)

38


39              j +=
1


40
#print iv.encode('hex'), c

41


42      print flag


43
```

```
if __name__ ==
"__main__":

44


main()
```

ps：脚本是看wp的时候拖来的，并不是自己写的Orz

RegExpert

考验正则的题目，做题目的时候由于对正则实在太不熟悉了，导致第一步都没有过，所以今天仔细研究下。

select

```
================= [SQL] =================

1


Please match
string
that
2 contains
"select"
as a case insensitive subsequence.
```

上来第一步是select，条件是必须正则匹配到所有包含select的子字符串，在select中的任意位置都可以插入任意字符。

于是当时我的初版正则是长这样的

```
[
Ss][
A-Za-z]?[
eE][
A-Za-z]?[
1 lL][
A-Za-z]?[
eE][
A-Za-z]?[
cC][
A-Za-z]?[tT]
```

当然是有长度限制的

```
(?i)s.
*e.*l.
1 *e.*c.
*t
```

递归正则？

```
1
```

```
2
```

实话实说，没有特别搞明白这个题的意思，大概是说递归语法?

这里应该需要用到ruby的语法

payload:

```
1
```

素数

```
1
```

```
2
```

这里需要强制所有元素为x，为了避免空的正则，所以我们需要 结尾

```
1
```

回文？

```
Both "QQ" and "TAT" are palindromes, but "PPAP" is not.
```

看上去应该同样是类似于递归的判断方式，取回文？我们需要匹配axa的模式，a为任意字符串模式

```
^(\w?|(\w)\g<1>\k<2>)$
```

或者

```
((.)(\g<1>)\2|.?)
```