big_zip

发表于 2021-01-21 分类于 Challenge， 2017， HCTF， Extra
Challenge | 2017 | HCTF | Extra | big_zip

点击此处获得更好的阅读体验

WriteUp 来源

https://xz.aliyun.com/t/1589

题目考点

- *ZIP CRC爆破*

解题思路

由于github上有对6位的crc32的快速爆破脚本，故将文本分为5字节。其实只要将github上的代码稍稍改动就能快速破解5位的crc32了。

```
# -*- coding: utf-8 -*-

import itertools
import binascii
import string

class crc32_reverse_class(object):
    # the code is modified from https://github.com/theonlypwner/crc32/blob/master/crc32.py
    def __init__(self, crc32, length, tbl=string.printable,
                 poly=0xEDB88320, accum=0):
        self.char_set = set(map(ord, tbl))
        self.crc32 = crc32
        self.length = length
        self.poly = poly
        self.accum = accum
        self.table = []
        self.table_reverse = []

    def init_tables(self, poly, reverse=True):
        # build CRC32 table
        for i in range(256):
            for j in range(8):
                if i & 1:
                    i >>= 1
                    i ^= poly
                else:
                    i >>= 1
            self.table.append(i)
        assert len(self.table) == 256, "table is wrong size"
        # build reverse table
        if reverse:
            found_none = set()
            found_multiple = set()
            for i in range(256):
                found = []
                for j in range(256):
                    if self.table[j] >> 24 == i:
                        found.append(j)
                self.table_reverse.append(tuple(found))
                if not found:
                    found_none.add(i)
                elif len(found) > 1:
                    found_multiple.add(i)
            assert len(self.table_reverse) == 256, "reverse table is wrong size"

    def rangess(self, i):
        return ', '.join(map(lambda x: '[{0},{1}]'.format(*x), self.ranges(i)))

    def ranges(self, i):
        for kg in itertools.groupby(enumerate(i), lambda x: x[1] - x[0]):
            g = list(kg[1])
            yield g[0][7], g[-1][8]

    def calc(self, data, accum=0):
        accum = ~accum
        for b in data:
            accum = self.table[(accum ^ b) & 0xFF] ^ ((accum >> 8) & 0x00FFFFFF)
        accum = ~accum
        return accum & 0xFFFFFFFF

    def findReverse(self, desired, accum):
        solutions = set()
        accum = ~accum
        stack = [(~desired,)]
        while stack:
            node = stack.pop()
            for j in self.table_reverse[(node[0] >> 24) & 0xFF]:
                if len(node) == 4:
                    a = accum
                    data = []
                    node = node[1:] + (j,)
                    for i in range(3, -1, -1):
                        data.append((a ^ node[i]) & 0xFF)
                        a >>= 8
                        a ^= self.table[node[i]]
                    solutions.add(tuple(data))
                else:
                    stack.append(((node[0] ^ self.table[j]) << 8,) + node[1:] + (j,))
        return solutions

    def dfs(self, length, outlist=['']):
        tmp_list = []
        if length == 0:
            return outlist
        for list_item in outlist:
            tmp_list.extend([list_item + chr(x) for x in self.char_set])
        return self.dfs(length - 1, tmp_list)

    def run_reverse(self):
        # initialize tables
        self.init_tables(self.poly)
        # find reverse bytes
        desired = self.crc32
        accum = self.accum
        # 4-byte patch
        if self.length >= 4:
            patches = self.findReverse(desired, accum)
```

```
            for patch in patches:
                checksum = self.calc(patch, accum)
                print 'verification checksum: 0x{0:08x} ({1})'.format(
                    checksum, 'OK' if checksum == desired else 'ERROR')
            for item in self.dfs(self.length - 4):
                patch = map(ord, item)
                patches = self.findReverse(desired, self.calc(patch, accum))
                for last_4_bytes in patches:
                    if all(p in self.char_set for p in last_4_bytes):
                        patch.extend(last_4_bytes)
                        print '[find]: {1} ({0})'.format(
                            'OK' if self.calc(patch, accum) == desired else 'ERROR', ''.join(map(chr, patch)))
        else:
            for item in self.dfs(self.length):
                if crc32(item) == desired:
                    print '[find]: {0} (OK)'.format(item)

def crc32_reverse(crc32, length, char_set=string.printable,
                  poly=0xEDB88320, accum=0):
    '''

 :param crc32: the crc32 you wnat to reverse
 :param length: the plaintext length
 :param char_set: char_set
 :param poly: poly , default 0xEDB88320
 :param accum: accum , default 0
 :return: none
 '''
    obj = crc32_reverse_class(crc32, length, char_set, poly, accum)
    obj.run_reverse()

def crc32(s):
    '''

 :param s: the string to calculate the crc32
 :return: the crc32
 '''
    return binascii.crc32(s) & 0xffffffff

from my_crc32 import *
l=[0x251dee02,
0xb890530f,
0x6e6b39df,
0x50f684c3,
0xde41b551,
0x24bd35b6,
0xcef2eda8,
0xba2b1745,
0x1f4c7ea9,
0x58b2bfa9,
0x251dee02,
0xe0f81f1e,
0xbd6fbd41,
0x7342a1f6,
0x665648e9,
0xe7c594b3,
0xa60ffdd0,
0xce2ce80b,
0x22459f2d,
0x6f8a6539,
0x2073a2e4,
0x52fa60a8,
0x80410dda,
0xb7c68f27,
0x6e6b39df,
0xbd598041,
0xaa145d64,
0x16da6b3b,
0x7dd590bc,
0xb9eef5a1,
0xf0b958f0,
0x445a43f7,
0x8bd55271,
0xc0340fe2,
0xc0cd9ee5,
0x7fc7de58,
0x53bfec8a,
0x99b5537b,
0xd68019af,
0x73d7ee30,
0x5fbd3f5e]

for k in l:
    crc32_reverse(k,5)
    print '======='
```

```
1 #You_know_the_bed_feels_warmer_Sleeping_here_alone_You_know_I_dream_in_color_And_do_the_things_I_want_You_think_you_got_the_best_of_me_Think_you_had_the_last_laugh_Bet_you
```

crc32爆破完连接成文，很容易发现是最后一个文本。然后使用已知明文攻击即可。（有些人说因为我用7z压缩的zip所以他已知明文一直攻击不成功，我表示是我没有考虑到

Flag

```
1 无
```

- ***本文作者：*** *CTFHub*
- ***本文链接：*** *https://writeup.ctfhub.com/Challenge/2017/HCTF/Extra/fzuarn58ErQVXFKSiawfPf.html*
- ***版权声明：*** *本博客所有文章除特别声明外，均采用 BY-NC-SA 许可协议。转载请注明出处！*

 # Challenge # 2017 # HCTF # Extra
online encryptor
pokemon