Scrambled-Eggs

发表于 2021-01-09 分类于 Challenge， 2020， CSICTF， Reverse
Challenge | 2020 | CSICTF | Reverse | Scrambled-Eggs

点击此处获得更好的阅读体验

WriteUp来源

https://dunsp4rce.github.io/csictf-2020/reversing/2020/07/22/Scrambled-Eggs.html

by `raghul-rajasekar`

题目描述

> *I like my eggs sunny side up, but I ended up scrambling them.*

题目考点

解题思路

Looking at `scrambledeggs.py`, there are several operations on the flag as well as the two keys (technically, `key2` is derived from `key1` so it's just one key) that we need to reverse. The important functions that need to be reversed are `enc1` and `enc2`:

```
map = ['v', 'r', 't', 'p', 'w', 'g', 'n', 'c', 'o', 'b', 'a', 'f', 'm', 'i', 'l', 'u', 'h', 'z', 'd', 'q', 'j', 'y', 'x', 'e', 'k', 's']
def enc1(text):
    n = random.randint(0,sys.maxsize%28)
    return text[n:] + text[:n]
def enc2(text):
    temp = ''
    for i in text:
        temp += map[ord(i)-ord('a')]
    return temp
```

`sys.maxsize%28` on my system was `7`, so I made the assumption that `n` would be in the range `[0, 7]`. `enc2` performs a simple substitution operation, which is simple to reverse. To reverse `enc1`, due to the randomness involved, we would have to brute-force through all 8 possible values of `n`. As there are 4 `enc1`s used, we would have to go through 8^4=4096 combinations to reverse all the `enc1`s.

There is also a swapping operation taking place, where the swapping within `flag`, `key1` and `key2` is all inter-dependent:

```
for j in range(2):
    for i in range(14):
        temp1 = flag[i]
        flag[i] = flag[(ord(key1[i])-ord('a'))%28]
        flag[(ord(key1[i])-ord('a'))%28] = temp1
        temp2 = key1[i]
        key1[i] = key1[(ord(key2[i])-ord('a'))%14]
        key1[(ord(key2[i])-ord('a'))%14] = temp2

    for i in range(14,28):
        temp1 = flag[i]
        flag[i] = flag[(ord(key2[i-14])-ord('a'))%28]
        flag[(ord(key2[i-14])-ord('a'))%28] = temp1
        temp2 = key2[i-14]
        key2[i-14] = key2[(ord(key1[i-14])-ord('a'))%14]
        key2[(ord(key1[i-14])-ord('a'))%14] = temp2
```

This is easy to reverse by reversing each swap individually. Care must be taken to run the indices in reverse order.

Next, the keys get swapped with 50% probability, so we have no choice but to try both possibilities. Finally, `key2` gets encrypted by adding it with another randomly generated string `k` modulo 26, to put it simply.

```
k = ''
for i in range(14):
    k += random.choice(map)
k = list(k)
key2 = k+key2
for i in range(14):
    a = ord(k[i])-ord('a')+ord(key2[i+14])
    if a>122:
        a=a%122
        a=a+97
    key2[i+14]= chr(a)
```

Note that if at the end of the i^{th} iteration, `chr(a) == k[i]`, then the original value of `key2[i+14]` could either have been `'a'` or `'z'`, so we'd have to account for this corner case at each position, possibly leading to an exponential increase in the number of candidates we'd have to check.

Combining all these observations together, we have this following code to reverse the flag:

```
orig_flag = 'lvvrafwgtocdrdzfdqotiwvrcqnd'
orig_key2 = 'eudlqgluduggdluqmocgyukhbqkx'
orig_key1 = 'xtfsyhhlizoiyx'
map = ['v', 'r', 't', 'p', 'w', 'g', 'n', 'c', 'o', 'b', 'a', 'f', 'm', 'i', 'l', 'u', 'h', 'z', 'd', 'q', 'j', 'y', 'x', 'e', 'k', 's']
maprev = [None] * 26
for i,n in enumerate(map):
    maprev[ord(n) - ord('a')] = chr(i + ord('a'))
def enc2rev(text):
    temp = ''
    for i in text:
        temp += maprev[ord(i)-ord('a')]
    return temp
key2 = orig_key2[:]
key1 = orig_key1[:]
key2 = enc2rev(key2)
key1, key2 = list(key1), list(key2)
k = key2[:14]
key2 = key2[14:]
candk2 = [[]]
# reversing key2 encryption
for i in range(14):
    x = ord(key2[i]) - ord(k[i])
    if x < 0:
        x = 25 + x
    x += 97
    if x == 97:
        candk2 = [cand + ['a'] for cand in candk2] + [cand + ['z'] for cand in candk2]
    else:
        candk2 = [cand + [chr(x)] for cand in candk2]
for key2 in candk2:
    # each such loop is used to reverse enc1
    for iter1 in range(8):
        flag1 = orig_flag[:]
        flag1 = flag1[-iter1:] + flag1[:-iter1]
        for iter2 in range(8):
            flag2 = flag1[:]
            flag2 = flag2[-iter2:] + flag2[:-iter2]
            for iter3 in range(8):
                flag3 = flag2[:]
                flag3 = flag3[-iter3:] + flag3[:-iter3]
                flag3 = enc2rev(flag3)
                flag3 = list(flag3)
                # reversing swapping of keys
                c = [(flag3, key1, key2),(flag3, key2, key1)]
                for flg, k1, k2 in c:
                    k1 = k1[:]
                    k2 = k2[:]
                    flg = flg[:]
                    for j in range(2):
                        for i in range(27, 13, -1):
                            temp2 = k2[i-14]
                            k2[i-14] = k2[(ord(k1[i-14])-ord('a'))%14]
                            k2[(ord(k1[i-14])-ord('a'))%14] = temp2
                            temp1 = flg[i]
                            flg[i] = flg[(ord(k2[i-14])-ord('a'))%28]
                            flg[(ord(k2[i-14])-ord('a'))%28] = temp1
                        for i in range(13, -1, -1):
                            temp2 = k1[i]
                            k1[i] = k1[(ord(k2[i])-ord('a'))%14]
                            k1[(ord(k2[i])-ord('a'))%14] = temp2
                            temp1 = flg[i]
                            flg[i] = flg[(ord(k1[i])-ord('a'))%28]
                            flg[(ord(k1[i])-ord('a'))%28] = temp1
                    for iter4 in range(8):
                        flg1 = flg[-iter4:] + flg[:-iter4]
                        flg1 = ''.join(flg1)
                        # check if flag format is found
                        if flg1[:6] == 'csictf':
                            print(flg1, k1, k2)
```

Finally, by replacing `'a'` *with braces and* `'b'` *with underscores appropriately, we get* `flag = 'csictf{all_the_kings_horses}'` *and* `key1 = 'together_again'`.

Flag

```
csictf{all_the_kings_horses}
```


 # Challenge # 2020 # Reverse # CSICTF
Vietnam
RicknMorty