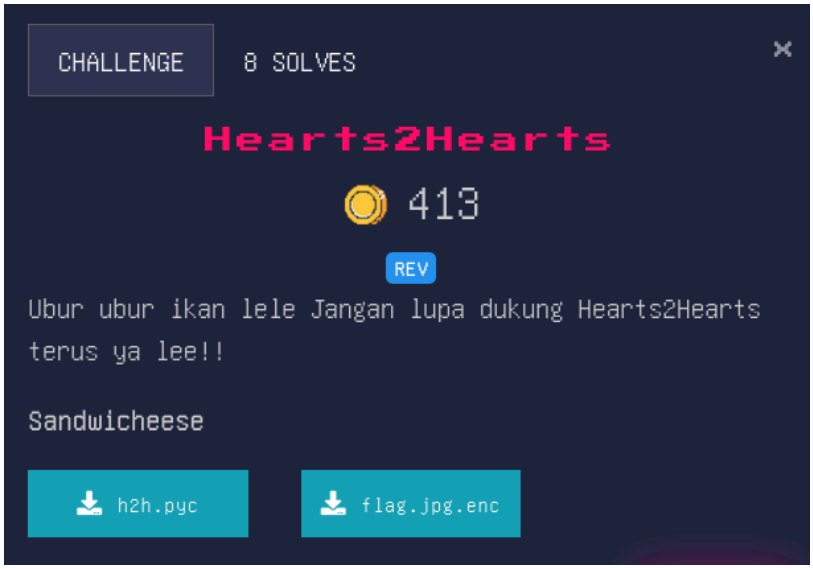
[Rev] Hearts2Hearts (413 points)

By: FieryBanana101



Diberikan sebuah file bytecode python ‘h2h.pyc’ dan gambar flag yang telah dienkripsi. Ide awal yang bisa kita asumsikan yaitu flag di enkripsi dengan menggunakan script yang telah di compile menjadi ‘h2h.pyc’. Akan digunakan tools ‘pycdas’ dari https://github.com/zrax/pycdc.

./arsenal/pycdc/pycdas -o dissass h2h.pyc

Hasil disassembly menunjukkan bahwa terdapat setidaknya terdapat 6 fungsi pada program, karena assembly dari vm python cukup intuitif, mari kita analisis dan pulihkan satu-persatu secara manual (hasil pemulihan dapat kita cross check dengan menggunakan modul ‘dis’).

lcg():

```
[Disassembly]
    0       RESUME                  0
    2       BUILD_LIST              0
    4       STORE_FAST              5: key
    6       LOAD_GLOBAL             1: NULL + range
   16       LOAD_FAST               4: size
   18       CALL                    1
   26       GET_ITER
   28       FOR_ITER                33 (to 96)
   32       STORE_FAST              6: _
   34       LOAD_FAST_LOAD_FAST     16: a, seed
   36       BINARY_OP               5 (*)
   40       LOAD_FAST               2: c
   42       BINARY_OP               0 (+)
   46       LOAD_FAST               3: m
   48       BINARY_OP               6 (%)
```

```
            52      STORE_FAST              0: seed
            54      LOAD_FAST               5: key
            56      LOAD_ATTR               3: append
            76      LOAD_FAST               0: seed
            78      LOAD_CONST              1: 256
            80      BINARY_OP               6 (%)
            84      CALL                    1
            92      POP_TOP
            94      JUMP_BACKWARD           35 (to 26)
            98      END_FOR
            100     POP_TOP
            102     LOAD_GLOBAL             5: NULL + bytes
            112     LOAD_FAST               5: key
            114     CALL                    1
            122     RETURN_VALUE
```

Sesuai namanya, ini adalah PRNG berbentuk *Linear Congruential Generator*. Fungsi ini menggunakan parameter initial value (seed), multiplier (a), increment(c) dan modulo (m). Hasil random number akan di modulo agar bisa dikonversi menjadi byte.

```
def lcg(seed, a, c, m, size):
    key = []
    for _ in range(size):
        seed = (a * seed + c) % m
        key.append(seed % 256)
    return bytes(key)
```

round_function:

```
                [Disassembly]
                0       RESUME                  0
                2       LOAD_FAST               0: data
                4       LOAD_CONST              1: 1
                6       BINARY_OP               3 (<<)
                10      LOAD_FAST               1: key
                12      BINARY_OP               12 (^)
                16      LOAD_CONST              2: 256
                18      BINARY_OP               6 (%)
                22      RETURN_VALUE
```

Fungsi ini hanya akan melakukan beberapa operasi pada parameter data dan key.

```
def round_function(data, key):
    return ((data << 1) ^ key) % 256
```

generate_iv:

```
            0       RESUME                  0
```

```
  2       LOAD_CONST                  1: <CODE> round_function
  4       MAKE_FUNCTION
  6       STORE_FAST                  2: round_function
  8       LOAD_FAST                   0: seed
 10       LOAD_CONST                  0: None
 12       LOAD_CONST                  2: 8
 14       BINARY_SLICE
 16       STORE_FAST                  3: left
 18       LOAD_FAST                   0: seed
 20       LOAD_CONST                  2: 8
 22       LOAD_CONST                  0: None
 24       BINARY_SLICE
 26       STORE_FAST                  4: right
 28       LOAD_GLOBAL                 1: NULL + range
 38       LOAD_FAST                   1: rounds
 40       CALL                        1
 48       GET_ITER
 50       FOR_ITER                    58 (to 168)
 54       STORE_FAST                  5: _
 56       LOAD_FAST                   4: right
 58       STORE_FAST                  6: new_left
 60       LOAD_GLOBAL                 3: NULL + bytes
 70       LOAD_GLOBAL                 1: NULL + range
 80       LOAD_CONST                  2: 8
 82       CALL                        1
 90       GET_ITER
 92       LOAD_FAST_AND_CLEAR         7: i
 94       SWAP                        2
 96       BUILD_LIST                  0
 98       SWAP                        2
100       GET_ITER
102       FOR_ITER                    19 (to 142)
106       STORE_FAST_LOAD_FAST        115: i, left
108       LOAD_FAST                   7: i
110       BINARY_SUBSCR
114       LOAD_FAST                   2: round_function
116       PUSH_NULL
118       LOAD_FAST_LOAD_FAST         71: right, i
120       BINARY_SUBSCR
124       LOAD_FAST                   7: i
126       CALL                        2
134       BINARY_OP                   12 (^)
138       LIST_APPEND                 2
140       JUMP_BACKWARD               21 (to 100)
```

```
            144     END_FOR
            146     POP_TOP
            148     SWAP                    2
            150     STORE_FAST              7: i
            152     CALL                    1
            160     STORE_FAST              8: new_right
            162     LOAD_FAST_LOAD_FAST     104: new_left, new_right
            164     STORE_FAST_STORE_FAST   67: right, left
            166     JUMP_BACKWARD           60 (to 48)
            170     END_FOR
            172     POP_TOP
            174     LOAD_FAST_LOAD_FAST     52: left, right
            176     BINARY_OP               0 (+)
            180     RETURN_VALUE
            182     SWAP                    2
            184     POP_TOP
            186     SWAP                    2
            188     STORE_FAST              7: i
            190     RERAISE                 0
```

Fungsi ini akan melakukan generasi IV yang akan digunakan pada enkripsi selanjutnya. Akan digunakan sebuah bytes ‘seed’ yang akan dibagi menjadi 2 bagian, left dan right dengan string slicing (line 8-24). Kemudian akan dilakukan for loop (line 28 - 170) sebanyak ‘round’ kali.

Di dalam for loop tersebut akan dilakukan perubahan nilai ‘left’ dan ‘right’ nilai ‘left’ baru yaitu nilai ‘right’. Sementara nilai ‘right’ baru yaitu sebuah bytes baru yang dibuat dari gabungan operasi xor dan round_function antara nilai sebelumnya (line 92-164). Hasil IV yaitu nilai left+right setelah iterasi selesai.

Note: saya tidak berhasil mencari berapa nilai ‘round’, namun dengan *trial and error* ditemukan bahwa nilainya yaitu 4

```
def generate_iv(seed):
   left = seed[:8]
   right = seed[8:]
   for _ in range(4):
       new_left = right
       new_right = bytes([left[i] ^ round_function(right[i], i) for i in range(8)])
       left, right = new_left, new_right
   return left+right
```

generate_key_and_iv:

```
[Disassembly]
            0       RESUME                  0
            2       LOAD_CONST              1: b'nyomanayucarmenita'
```

```
              4       STORE_FAST                  0: seed
              6       LOAD_GLOBAL                 1: NULL + sum
              16      LOAD_FAST                   0: seed
              18      CALL                        1
              26      STORE_FAST                  1: lcg_seed
              28      LOAD_GLOBAL                 3: NULL + lcg
              38      LOAD_FAST                   1: lcg_seed
              40      LOAD_CONST                  2: 1664525
              42      LOAD_CONST                  3: 1013904223
              44      LOAD_CONST                  4: 0x100000000
              46      LOAD_CONST                  5: 16
              48      LOAD_CONST                  6: ('a', 'c', 'm', 'size')
              50      CALL_KW                     5
              52      STORE_FAST                  2: key
              54      LOAD_GLOBAL                 5: NULL + generate_iv
              64      LOAD_FAST                   0: seed
              66      CALL                        1
              74      STORE_FAST                  3: iv
              76      LOAD_GLOBAL                 7: NULL + print
              86      LOAD_FAST_LOAD_FAST         35: key, iv
              88      CALL                        2
              96      POP_TOP
              98      LOAD_FAST_LOAD_FAST         35: key, iv
              100     BUILD_TUPLE                 2
              102     RETURN_VALUE
```

Fungsi ini akan melakukan generasi iv dan key menggunakan beberapa nilai konstanta yang telah ditentukan. Digunakan juga fungsi-fungsi sebelumnya. Key dihasilkan dari lcg(), dan IV dihasilkan dari generate_iv(). Hasilnya akan di print lalu di return.

```
def generate_iv_and_key():
    seed = b'nyomanayucarmenita'
    lcg_seed = sum(seed)
    key = lcg(seed=lcg_seed, a=1664525, c=1013904223, m=0x100000000,size=16)
    iv = generate_iv(seed)
    print(key, iv)
    return key, iv
```

encrypt_flag:

```
          [Disassembly]
              0       RESUME                      0
              2       LOAD_GLOBAL                 1: NULL + generate_key_and_iv
              12      CALL                        0
              20      UNPACK_SEQUENCE             2
              24      STORE_FAST_STORE_FAST       1: key, iv
              26      LOAD_GLOBAL                 2: AES
              36      LOAD_ATTR                   4: new
```

```
             56      PUSH_NULL
             58      LOAD_FAST                0: key
             60      LOAD_GLOBAL              2: AES
             70      LOAD_ATTR                6: MODE_CBC
             90      LOAD_FAST                1: iv
             92      CALL                     3
             100     STORE_FAST               2: cipher
             102     NOP
             104     LOAD_GLOBAL              9: NULL + open
             114     LOAD_CONST               1: 'flag.jpg'
             116     LOAD_CONST               2: 'rb'
             118     CALL                     2
             126     BEFORE_WITH
             128     STORE_FAST               3: f
             130     LOAD_FAST                3: f
             132     LOAD_ATTR                11: read
             152     CALL                     0
             [truncated]
```

Fungsi ini hanya akan melakukan enkripsi pada file ‘flag.jpg’ dengan menggunakan AES block cipher CBC. Key dan IV yang digunakan berasal dari fungsi generate_key_iv().

```
def encrypt_flag():
   key, iv = generate_key_and_iv()
   with open('flag.jpg', 'rb') as f:
       encrypted_flag = AES.new(key, AES.MODE_CBC, iv).encrypt(f.read())

   with open('flag.jpg.enc', 'rb') as f:
       f.write(encrypt_flag)
```

main():

```
          [Disassembly]
              0       RESUME                   0
              2       LOAD_GLOBAL              1: NULL + encrypt_flag
              12      CALL                     0
              20      POP_TOP
              22      RETURN_CONST             0: None
```

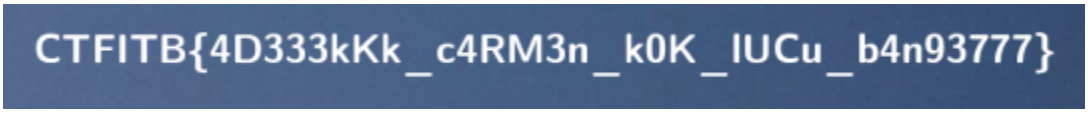
Terakhir, fungsi main() hanya sebagai wrapper yang akan memanggil encrypt_flag()

```
def main():
   encrypt_flag()

if __name__ == "__main__":
   main()
```

Dengan menggabungkan seluruh fungsi yang telah dipulihkan dan sedikit modifikasi, dapat kita lakukan pemulihan ‘flag.jpg’ dengan solver script berikut.

solve.py:

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

def lcg(seed, a, c, m, size):
    key = []
    for _ in range(size):
        seed = (a * seed + c) % m
        key.append(seed % 256)
    return bytes(key)

def round_function(data, key):
    return ((data << 1) ^ key) % 256

def generate_iv(seed):
    left = seed[:8]
    right = seed[8:]
    for _ in range(4):
        new_left = right
        new_right = bytes([left[i] ^ round_function(right[i], i) for i in range(8)])
        left, right = new_left, new_right
    return left+right

def generate_iv_and_key(): #
    seed = b'nyomanayucarmenita'
    lcg_seed = sum(seed)
    key = lcg(seed=lcg_seed, a=1664525, c=1013904223, m=0x100000000,size=16)
    iv = generate_iv(seed)
    print(key, iv)
    return key, iv

key, iv = generate_iv_and_key()

with open("flag.enc", "rb") as f:
    content = unpad(AES.new(key, AES.MODE_CBC, iv).decrypt(f.read()), 16)

with open("flag.jpg", "wb") as f:
    f.write(content)

import os
os.system("xdg-open flag.jpg")
```



FLAG: CTFITB{4D333kKk_c4RM3n_k0K_lUCu_b4n93777}