WriteUp Cyber Jawara 2021

I Putu Belugayasa Anesa Putra

lMaddiXl
ChaO
AnehMan

Cryptography

1. Shuvvler

a. Executive Summary

Notice that shuvvler is the next gen of shuffler.

Wrap your flag with CJ2021{...} format.

Author: deomkicer#3362

b. Technical Report

Diberikan file chall.py dan flag.html.enc. Berikut penampakannya
chall.py

```
#!/usr/bin/env python3
import random, string, sys

chars = string.ascii_lowercase + string.ascii_uppercase +
string.digits + string.punctuation

def func1(s, key):
    r = ''
    for i in range(len(s)):
        if s[i] in chars:
            if   s[i] in string.ascii_lowercase: z =
string.ascii_lowercase
            elif s[i] in string.ascii_uppercase: z =
string.ascii_uppercase
            elif s[i] in string.digits:          z =
string.digits
            else:                                z =
string.punctuation
            r += z[(z.index(s[i]) + key) % len(z)]
        else:
            r += s[i]
    return r

def func2(s):
```

```
    return     s.translate(s.maketrans(chars,     chars[13:]     +
chars[:13]))

def encrypt(s):
    key = random.getrandbits(128)
    r = s
    for _ in range(8):
        r = func1(r, key)
        r = func2(r)
    return r

def main():
    fns = sys.argv[1:]
    for fn in fns:
        try:
            x = open(fn, 'r').read()
            f = open(fn + '.enc', 'w')
            f.write(encrypt(x))
            f.close()
        except:
            continue

if __name__ == '__main__':
    if len(sys.argv) < 2:
        print(f'Usage: {sys.argv[0]} <file>')
        print(f'Example: {sys.argv[0]} flag.html')
        sys.exit()
    main()
```

Flag.html.enc (potongan)

```
*&rkj,
  *`gt _jyqq+:qry!=:,
    *`gt _jyqq+:jyw=P:,*6`gt,
    *`gt _jyqq+:jyw=P:,*6`gt,
    *`gt _jyqq+:jyw=P:,*6`gt,
    *`gt _jyqq+:jyw=P:,*6`gt,
    *`gt _jyqq+:jyw=P:,*6`gt,
    *`gt _jyqq+:jyw=P:,*6`gt,
```

Intinya script melakukan enkripsi mirip seperti vigenere cipher + monoalphabetic substitution. Dilakukan 8 kali proses enkripsi dengan key

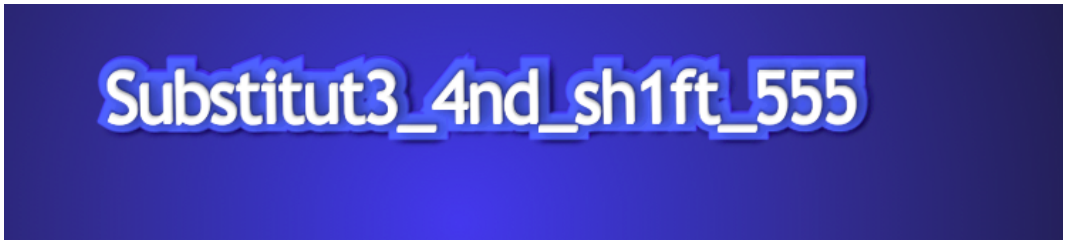
yang sama dan random (128-bit). Jadi tinggal menukar urutan, dan balik `func1` dan `func2`. Karena key nantinya akan di-modulus dengan panjang charset (tergantung jenis karakter), dan panjang charset tidak sama semua, jadi kita tidak bisa mencoba brute sampai total panjang charset. Jadi saya mencoba brute dari 0-10000, dan untuk mempercepat proses brute, kami hanya mencoba decrypt line pertama dari ciphertext, karena sudah pasti string pada line pertama adalah `<html>`

Ketika string yang diinginkan sudah didapat, maka tinggal gunakan key yang sama untuk decrypt keseluruhan ciphertext. Berikut full scriptnya

```
import string

chars = string.ascii_lowercase + string.ascii_uppercase +
string.digits + string.punctuation

def decrypt(s, key):
    def func1(s, key):
        r = ''
        for i in range(len(s)):
            if s[i] in chars:
                if   s[i] in string.ascii_lowercase: z =
string.ascii_lowercase
                elif s[i] in string.ascii_uppercase: z =
string.ascii_uppercase
                elif s[i] in string.digits:          z =
string.digits
                else:                                z =
string.punctuation
                r += z[(z.index(s[i]) - key) % len(z)]
            else:
                r += s[i]
        return r

    def func2(s):
        return s.translate(s.maketrans(chars[13:] + chars[:13],
chars))

    r = s
    for _ in range(8):
        r = func2(r)
        r = func1(r, key)
```

```
    return r

flag_enc = open("flag.html.enc").read()
final_key = {}
line1 = flag_enc.split("\n")[0]

for k in range(10000):
    res = decrypt(line1, k)
    if res == "<html>":
        flag = decrypt(flag_enc, k)
        with open("result.html", "w") as f:
            f.write(flag)
        break
```

Hasil:



c. Flag

Flag: **CJ2021{Substitut3_4nd_sh1ft_555}**

Reverse Engineering

1. crac

a. Executive Summary

Can you flag this crac?

Author: vidner#6838

b. Technical Report

Diberikan file ELF 64-bit not stripped. Berikut penampakannya di IDA

```
puts("crac?");
__isoc99_scanf("%s", s);
if ( (unsigned int)strlen(s) == 44 )
{
  for ( i = 0; i < 44; ++i )
  {
    v6 = 0;
    for ( j = 0; j <= i; ++j )
      v6 += crac(&s[j], 1LL);
    if ( v7[i] != v6 )
    {
      puts("Incorrect!");
      return 0;
    }
  }
  puts("Correct!");
  result = 0;
}
else
{
  puts("Incorrect!");
  result = 0;
}
return result;
```

Sebenarnya lumayan sederhana. Program mengecek inputan user apakah panjang karakter sama dengan 44, lalu karakter tersebut dimasukkan ke fungsi `crac`, dan di cek satu persatu. Tapi ketika di-translate ke python, entah kenapa tidak berhasil. Jadi kami solve manual dengan GDB (sebenarnya bisa dibuat otomatis pake GDB script, cuma gaada yg ngerti akwoakowkoakow)

Berikut penampakan disas GDB

```
0x0000555555554a3d <+615>:   call   0x55555555476a <crac>
0x0000555555554a42 <+620>:   mov    edx,eax
0x0000555555554a44 <+622>:   mov    eax,DWORD PTR [rbp-0x1c8]
0x0000555555554a4a <+628>:   add    eax,edx
0x0000555555554a4c <+630>:   mov    DWORD PTR [rbp-0x1c8],eax
0x0000555555554a52 <+636>:   add    DWORD PTR [rbp-0x1cc],0x1
0x0000555555554a59 <+643>:   mov    eax,DWORD PTR [rbp-0x1cc]
0x0000555555554a5f <+649>:   cmp    eax,DWORD PTR [rbp-0x1d0]
0x0000555555554a65 <+655>:   jle    0x555555554a23 <main+589>
0x0000555555554a67 <+657>:   mov    eax,DWORD PTR [rbp-0x1d0]
0x0000555555554a6d <+663>:   cdqe
0x0000555555554a6f <+665>:   mov    edx,DWORD PTR [rbp+rax*4-0x1c0]
0x0000555555554a76 <+672>:   mov    eax,DWORD PTR [rbp-0x1c8]
0x0000555555554a7c <+678>:   cmp    edx,eax
0x0000555555554a7e <+680>:   je     0x555555554a93 <main+701>
0x0000555555554a80 <+682>:   lea    rdi,[rip+0xf6]        # 0x555555554b7d
0x0000555555554a87 <+689>:   call   0x555555554610 <puts@plt>
0x0000555555554a8c <+694>:   mov    eax,0x0
0x0000555555554a91 <+699>:   jmp    0x555555554ad0 <main+762>
0x0000555555554a93 <+701>:   add    DWORD PTR [rbp-0x1d0],0x1
0x0000555555554a9a <+708>:   mov    eax,DWORD PTR [rbp-0x1d0]
0x0000555555554aa0 <+714>:   cmp    eax,DWORD PTR [rbp-0x1c4]
0x0000555555554aa6 <+720>:   jl     0x555555554a0d <main+567>
0x0000555555554aac <+726>:   lea    rdi,[rip+0xd5]        # 0x555555554b88
0x0000555555554ab3 <+733>:   call   0x555555554610 <puts@plt>
0x0000555555554ab8 <+738>:   mov    eax,0x0
```

Jika dibandingkan dengan hasil decompile IDA, kita set breakpoint di 0x0000555555554a7c (if kedua). Jika program lanjut, maka input benar. Jika program exit, maka input salah.
Setelah beberapa saat, ditemukan bahwa karakter pertama adalah huruf "C"

```
[-------------------------------------code-------------------------------------]
   0x555555554a6d <main+663>:   cdqe
   0x555555554a6f <main+665>:   mov    edx,DWORD PTR [rbp+rax*4-0x1c0]
   0x555555554a76 <main+672>:   mov    eax,DWORD PTR [rbp-0x1c8]
=> 0x555555554a7c <main+678>:   cmp    edx,eax
   0x555555554a7e <main+680>:   je     0x555555554a93 <main+701>
   0x555555554a80 <main+682>:   lea    rdi,[rip+0xf6]        # 0x555555554b7d
   0x555555554a87 <main+689>:   call   0x555555554610 <puts@plt>
   0x555555554a8c <main+694>:   mov    eax,0x0
```

```
RAX: 0x77cde91a
RBX: 0x0
RCX: 0xffffff00
RDX: 0x77cde91a
```

Jadi, bisa dipastikan bahwa string awal adalah `CJ2021{`
Sekarang tinggal mencari lanjutannya. Dengan melihat konteks, kita bisa mendapatkan kata pertama yaitu "`this`", dan spasi diganti dengan underscore ("_"). Untungnya, flag tidak menggunakan bahasa 1337 sehingga mempermudah proses brute manual. Makasi banyak min :')
Lanjutkan terus sampai flag didapat.

```
RAX: 0x83e7c438
RBX: 0x0
RCX: 0xffffff00
RDX: 0x83e7c438
RSI: 0xff
RDI: 0x7fffffffdc1c ("o", 'A' <repeats 30 times>, "}")
```

```
RAX: 0x620b9d61
RBX: 0x0
RCX: 0xffffff00
RDX: 0x620b9d61
RSI: 0xff
RDI: 0x7fffffffdc1d ("n", 'A' <repeats 29 times>, "}")
```

```
RAX: 0x5773afa9
RBX: 0x0
RCX: 0xffffff00
RDX: 0x5773afa9
RSI: 0xff
RDI: 0x7fffffffdc1e ("e", 'A' <repeats 28 times>, "}")
```

Ditemukan kata “one”, dan dilanjutkan dengan underscore. Dibawah adalah hasil mencari kata selanjutnya

```
RAX: 0xa43dd41
RBX: 0x0
RCX: 0xffffff00
RDX: 0xa43dd41
RSI: 0xff
RDI: 0x7fffffffdc20 ("s", 'A' <repeats 26 times>, "}")
```

```
RAX: 0xcee971dc
RBX: 0x0
RCX: 0xffffff00
RDX: 0xcee971dc
RSI: 0xff
RDI: 0x7fffffffdc21 ("h", 'A' <repeats 25 times>, "}")
```

```
RAX: 0xa9cc367a
RBX: 0x0
RCX: 0xffffff00
RDX: 0xa9cc367a
RSI: 0xff
RDI: 0x7fffffffdc22 ("o", 'A' <repeats 24 times>, "}")
```

Ditemukan string “sho”. Kata dalam bahasa Inggris yang diawali dengan “sho” adalah “should”, dan berdasarkan konteks yang ada, kemungkinan kata selanjutnya adalah “be”, sehingga konten flag yang didapat adalah “this_one_should_be_”. Berikut lanjutannya

```
RAX: 0x2031aaf1
RBX: 0x0
RCX: 0xffffff00
RDX: 0x2031aaf1
RSI: 0xff
RDI: 0x7fffffffdc2a ("e", 'A' <repeats 16 times>, "}")
```

```
RAX: 0x69e0f85
RBX: 0x0
RCX: 0xffffff00
RDX: 0x69e0f85
RSI: 0xff
RDI: 0x7fffffffdc2b ("a", 'A' <repeats 15 times>, "}")
```

Ditemukan string “ea”. Dengan melihat konteks, ditemukan kata selanjutnya adalah “easy”. Dengan sisa 15 karakter yang tidak diketahui, kemungkinan ada lagi 2 kata yang perlu dicari. Tapi dengan melihat konten flag “this_should_be_easy_”, kita mencoba melanjutkan kalimat dengan menambahkan “enough_right” karena terdengar masuk akal. Kita coba inputkan, ternyata benar

```
crac?
CJ2021{this_one_should_be_easy_enough_right}
Correct!
```

c. Flag

Flag: **CJ2021{this_one_should_be_easy_enough_right}**