

## 4CR - 200 pts - Reversing

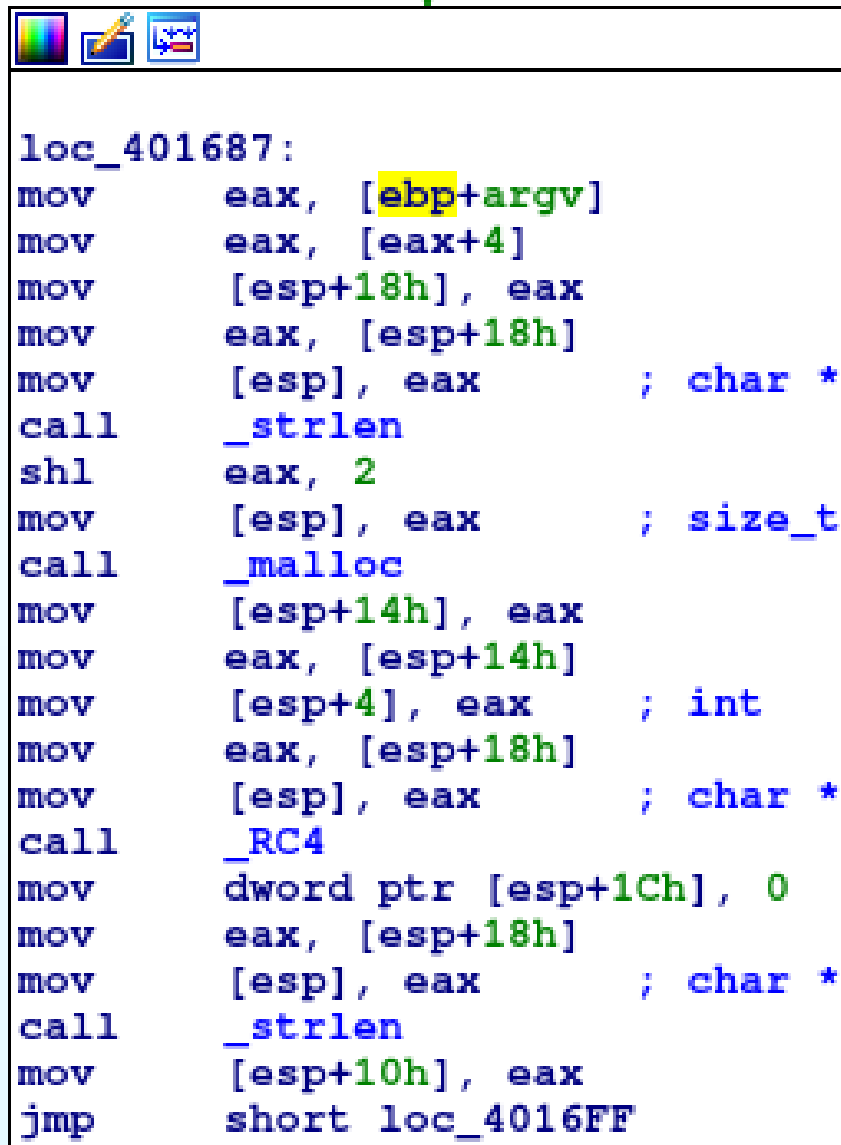

La descripción del reto nos dice lo siguiente:

**Analiza el binario C2C\_module.exe adjuntado, y ayudanos a desencriptar el archivo "secret" para obtener la flag.**

Lo primero que nos hace pensar el reto por su nombre es que se trata de un cifrado por **RC4**. (<https://es.wikipedia.org/wiki/RC4>)

Al revisar el desensamblado podemos confirmar nuestras sospechas.

Aquí podemos ver como llama a una función **RC4**:



```
loc_401687:
mov     eax, [ebp+argv]
mov     eax, [eax+4]
mov     [esp+18h], eax
mov     eax, [esp+18h]
mov     [esp], eax      ; char *
call    _strlen
shl     eax, 2
mov     [esp], eax      ; size_t
call    _malloc
mov     [esp+14h], eax
mov     eax, [esp+14h]
mov     [esp+4], eax    ; int
mov     eax, [esp+18h]
mov     [esp], eax      ; char *
call    _RC4
mov     dword ptr [esp+1Ch], 0
mov     eax, [esp+18h]
mov     [esp], eax      ; char *
call    _strlen
mov     [esp+10h], eax
jmp     short loc_4016FF
```

La cual llama a otras dos de nombre KSA y PRGA:

```

lea     edx, [eax-1]
mov     [ebp+var_C], edx
mov     edx, eax
mov     eax, 10h
sub     eax, 1
add     eax, edx
mov     ecx, 10h
mov     edx, 0
div     ecx
imul    eax, 10h
call    ___chkstk_ms
sub     esp, eax
lea     eax, [esp+0Ch]
add     eax, 0
mov     [ebp+var_10], eax
mov     eax, [ebp+var_10]
mov     [esp], eax
call    _KSA
mov     eax, [ebp+var_10]
mov     edx, [ebp+arg_4]
mov     [esp+8], edx      ; int
mov     edx, [ebp+arg_0]
mov     [esp+4], edx      ; char *
mov     [esp], eax        ; int
call    _PRGA
mov     eax, 0
mov     esp, ebx
mov     ebx, [ebp+var_4]
leave
retn

```

Lo unico que queda es revisar **KSA** en busca de la llave utilizada.

```
; Attributes: bp-based frame

public _KSA
_KSA proc near

var_1C= dword ptr -1Ch
var_18= dword ptr -18h
var_14= dword ptr -14h
var_10= dword ptr -10h
var_C= dword ptr -0Ch
arg_0= dword ptr 8

push    ebp
mov     ebp, esp
sub     esp, 38h
mov     [ebp+var_18], offset aHackdefCommand ; "hackdef_command&control_key"
mov     eax, [ebp+var_18]
mov     [esp], eax ; char *
call    _strlen
mov     [ebp+var_1C], eax
mov     [ebp+var_C], 0
mov     [ebp+var_10], 0
jmp     short loc_401476
```

Con lo que llave utilizada es `hackdef_command&control_key`

Finalmente, tomamos una implementación de RC4 que nos arrojó la flag

```
$ ./script.py
hackdef{RC4_is_c0mm0nly_used_by_m4lw4r3}
$
```

Aquí el script utilizado para descifrar el archivo secret:

```
#!/usr/bin/python
import sys

def KSA(key):
    keylen = len(key)

    S = range(0x100)
    j = 0
    for i in range(0x100):
        j = (j + S[i] + key[i % keylen]) % 0x100
        S[i], S[j] = S[j], S[i] # swap
    return S

def PRGA(S):
```

```

i = 0
j = 0
while True:
    i = (i + 1) % 0x100
    j = (j + S[i]) % 0x100
    S[i], S[j] = S[j], S[i] # swap
    yield S[(S[i] + S[j]) % 0x100]

key = "hackdef_command&control_key"
key = [ord(x) for x in key]
S = KSA(key)

ciphertext = open("./secret").read().decode("hex")

keystream = PRGA(S)

for x in ciphertext:
    sys.stdout.write(chr(ord(x) ^ keystream.next()))
sys.stdout.write("\n")

```