**HW0 Writeup**

**R10922043 黃政瑋**

**CTF account: cwhuang1937**

1. **to bf or not to bf**

思路:

由於這兩張照片加密時所用的seed是一樣的，所以他們都是對同一組數字的序列做XOR，但當初加密時的時間並無從得知，所以沒辦法回推那個加密的序列是什麼。後來想到XOR有**交換律**，因此試著將這兩張照片做XOR，得到的結果仍會是自己，結果做完運算後的結果印出來就是答案了。

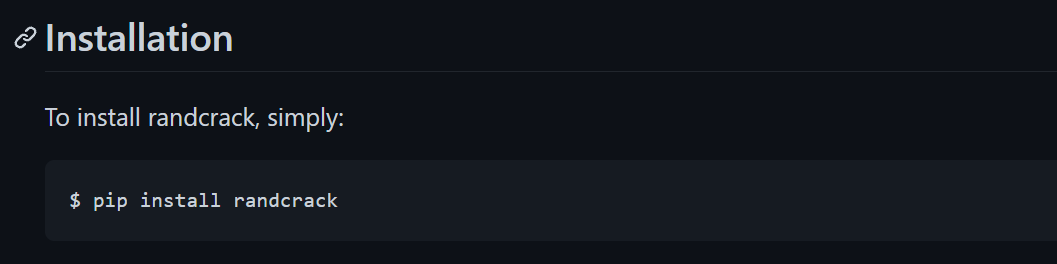
後來上了10/1的課才明白原來出題是用兩張不同的照片，所以兩張加密後的照片做XOR就像是將這兩張照片疊在一起的結果。

一張含有 文字 的圖片

自動產生的描述

題外話:

在這之前有先試著研究python random的實作(Mersenne Twister)，並找到一個cracker(<https://github.com/tna0y/Python-random-module-cracker>)，但與這題的情形無關，故就沒有再往這個方向嘗試了。



1. **XAYB**

思路:

首先用IDA Freeware來查看Pseudocode，可以發現這題的數字答案每次都是random的，所以不可能猜出來，但可以發現真的猜對時會印出一個字串，因此先用objdump查看assembly code後，再用gdb將斷點設在判斷處(仔細找cmp相關且是跟0x5比較的)，接著將$rip的值直接設在該if裡面，繼續執行即可印出flag。

一張含有 文字 的圖片

自動產生的描述

一張含有 文字, 電池, 計分板, 匾額 的圖片

自動產生的描述

1. **Arch Check**

思路:

首先用IIDA Freeware來查看Pseudocode，可以看到我們輸入進去的buffer只有32bytes，又加上這題沒有阻擋輸出超過32bytes的情況，因此可以試著用buffer overflow來攻擊。接著用objdump查看，可以發現有個debug function裡面藏有一個開shell的指令，因此將該指令的address記錄下來(即下圖的0x4011dd)。接著用python的pwntools來連到remote，並塞入適當的payload來使main的return address為上述紀錄的address即可。

Payload的計算:

宣告完buffer時的stack從上到下依序為local variables、$rbp、return address，因此buffer的32bytes加上$rbp的8bytes即等於40bytes，在這40bytes後的的位置即是return address所在的地方。

一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述

1. **text2emoji**

思路:

仔細觀察soucre那邊的原始碼，可以發現/looksLikeFlag這個路徑是跟flag相關的，但輸入那邊正常輸入都會走/emoji/:text這裡，因此要想辦法在輸入端讓他轉到我們要的路徑。但由於request過去的字串會先被檢查一次，當中又阻擋了”.”這個符號，因此這邊我卡關很久。後來在與兩位同學吳添毅(台大)、洪邵澤(交大)討論下，查到了**url encoding**，才成功將”.”轉換成**%2e**，此時即可成功切換路徑到/looksLikeFlag下。但這時得到的回傳結果是undefined，才發現他傳回來的結果是result而不是我們要的looksLikeFlag:true/false，再加上這題只會回傳包含或不包含的結果，所以我用python來暴力發request，直到looksLikeFlag為true，並是以”}”這個結尾，此時即可得到flag。

一張含有 文字 的圖片

自動產生的描述