CS: hw6 - break

學號：b04902053 姓名：鄭淵仁

flag:

```
CTF{PinADXAnInterfaceforCustomizableDebuggingwithDynamicInstrumentation}
```

write-up

reverse binary 檔

首先 reverse `break` 。

reverse 完之後，我發現 `break` 會把 input 的 string 一個 char 一個 char 照順序計算正確與否。而如果發現計算的結果是錯誤的，就會馬上 `puts` 出錯誤訊息然後 `return 0` 。而且每次 `break` 計算正確與否的過程都會花特別多的 instruction 。

所以可以透過觀察 instruction 的數量多寡，來判斷這次前幾個 char 是正確的，也就可以一個 char 一個 char 的拼出 flag。

另外，從 assembly 中我還發現他只會比較 `0x48` 個 byte ，所以只要拼出 `0x48` 個 char 就可以比對出正確的 flag 了。

使用 `pintool` 觀察 instruction 數量

我使用的方法是從第一個 char 開始一個一個往後猜，而每次要猜的 char 都是跑過所有的 printable character。

而猜法就是把要猜的 char 接到以前猜到的 string 的後面，然後跑一遍 `break` ，記下 instruction 數量。

等跑過所有的 printable character 之後，從裡面挑出 instruction 數量最多的，並且也應該是這個數量的 instruction 唯一的一組解，接到 string 的後面，再繼續猜下一個 char。

就這樣一直重複上述步驟，直到程式 `puts` 出 `Pass` 為止。

script

- 可以使用 `pintool` 觀察 instruction 數量並找出 flag 的 `python3` script： `break.py`
- 使用方法：
  - 我使用 `pintool` 的 `ManualExamples` 裡面的 `inscount0` 來計算 instruction 數量，所以必須先 compile 他，compile 方法如下：

    ```
    cd pin/source/tools/ManualExamples
    make obj-intel64/inscount0.so
    ```

- compile 完之後，把 `break.py` 放在跟 `break` 同一層資料夾下。把 `pintool` 的路徑用 `argv` 給他，就可以執行 `break.py` 。

```
python3 break.py [path/to/pintool]
# e.g.:
# python3 break.py ~/download/pin
```