計算機安全 HW7 Writeup

- Real name: 張瀚文
- Nickname on course website: Hwww
- Student ID: b07505027.

心得

Going Crazy

解法

這題的binary被strip過了，因此丟到ida會顯示不出函數名，需要用別人寫過的程式來復原，以便逆向。這邊用了ida7.0，在golang_loader_assist (https://github.com/strazzere/golang_loader_assist)找到了一個版本是可以在7.0上面執行的，使用後就可以找到 `main.main` 跟 `main.check_input` 等函數，並且還有幾個從名字上看不出的函數，如 `main.bezu`

不過就算可以復原名稱，用ida看go還是很奇怪，除了每個函數都會有個多餘的 `runtime_morestack` 之外，在go中的一個資料型態在ida裡面可能會被拆成好幾種，如slice就被切成ptr, len, cap三個變數，讀起來需要特別注意跟猜測，也配合自己編譯一些簡單的go程式來看看函數在ida中會長怎樣。

後來用ida看組語跟ida編出來的C code大概可以看出來，程式會檢查輸入字串的前後是不是 `x` ，之後把 `x` 給trim掉，再來以 `,` 來分割輸入的字串，並把分割的結果（一個slice）丟入 `main.check_input` 裡檢查。

在 `main.check_input` 裡面，會把分割出來的結果（假設叫 `arr` ），依照一個預先存好的順序（indexes），經過 `main.bezu` 這個看起來是某種加密函數後，跟另一個預先存好的陣列比較，若相同則通過（要求 bezu(arr[index]) == ans for index in indexes ），共比較36次。所以我們只要送出 `x{arr[0]},{arr[1]}, ... ,{arr[i]}, ... ,{arr[35]}x` 讓它通過檢查就好了。

預先存好的順序陣列indexes可以用gdb看，如指令 `x/36dg 0x4D23C0`

在湊出arr之前，有試著直接改register讓檢查通過，但只通過檢查並不會輸出flag，也把所有會輸出的地方都跑過了，所以合理推測flag應該就是由 `arr` 組成的。

雖然我們很難看出來bezu裡到底在做什麼，但經過試誤跟簡單的暴搜後發現，當 `arr[i]` 是空字串或是0的時候，程式會輸出 `No!!!That's way too crazy!!Stop!!` ，若 `arr[i]` 是正確答案的話，雖然不會顯示說 `arr[i]` 猜對了，但因為這時候程式會去看 `arr[next_index]` ，而如果 `arr[next_index]` 是空字串的話，就會輸出 `No!!!That's way too crazy!!Stop!!` ，也就是說可以間接驗證 `arr[i]` 有沒有猜對。藉由這點，就可以暴搜出整個 `arr` 了～程式碼可以參考 `code/hack.py`

關於驗證暴搜的部分本來也想直接用gdb看register有沒有相等，但因為不會用pwntools跟gdb自動化，所以就沒有試了。