**Write-Up: Reverse Engineering & Cracking the Challenge**

**1. Phân tích chương trình**

Chương trình kiểm tra một **mật khẩu bí mật** mà người dùng nhập vào. Nếu mật khẩu sau khi được **biến đổi** khớp với một chuỗi cố định (output), chương trình in ra **flag**.

**🔹 Điểm chính trong mã nguồn**

* **Chuỗi đầu ra (output) đã mã hóa**:

c

Sao chépChỉnh sửa

builtin\_strncpy(output,"apijaczhzgtfnyjgrdvqrjbmcurcmjczsvbwgdelvxxxjkyigy",0x33);

Đây là giá trị mà **mật khẩu phải trùng khớp** sau khi trải qua thuật toán mã hóa.

* **Nhận mật khẩu từ người dùng**:

c

Sao chépChỉnh sửa

\_\_isoc99\_scanf(&DAT\_00402024,input);

* + Người dùng nhập một chuỗi mật khẩu (input).
* **Mã hóa mật khẩu 3 lần**:

c

Sao chépChỉnh sửa

for (i = 0; i < 3; i++) {

for (j = 0; j < (int)opLength; j++) {

uVar1 = (j % 0xff >> 1 & 0x55U) + (j % 0xff & 0x55U);

uVar1 = ((int)uVar1 >> 2 & 0x33U) + (uVar1 & 0x33);

iVar2 = ((int)uVar1 >> 4) + input[j] + -0x61 + (uVar1 & 0xf);

input[j] = (char)iVar2 + (char)(iVar2 / 0x1a) \* -0x1a + 'a';

}

}

* + Mỗi ký tự của input **bị thay đổi** dựa trên index j.
  + **Lặp lại 3 lần**, khiến giá trị thay đổi dần dần.
* **So sánh kết quả với output**:

c

Sao chépChỉnh sửa

iVar2 = memcmp(input,output,(long)(int)opLength);

if (iVar2 == 0) {

printf("SUCCESS! Here is your flag: %s ","picoCTF{sample\_flag}");

}

* + Nếu input sau khi mã hóa **giống output**, chương trình in **flag**.
  + Nếu không, in **"FAILED!"**.

**2. Phân tích thuật toán mã hóa**

Chúng ta cần hiểu cách **mật khẩu bị thay đổi** qua mỗi vòng lặp.

**Bước 1: Tính uVar1 từ vị trí j**

c

Sao chépChỉnh sửa

uVar1 = (j % 0xff >> 1 & 0x55U) + (j % 0xff & 0x55U);

uVar1 = ((int)uVar1 >> 2 & 0x33U) + (uVar1 & 0x33);

* Tạo số uVar1 dựa vào **index j của ký tự**.
* Sử dụng **dịch bit (>>), phép AND (&)** để tạo **giá trị thay đổi** theo vị trí.

**Bước 2: Điều chỉnh giá trị input[j]**

c

Sao chépChỉnh sửa

iVar2 = ((int)uVar1 >> 4) + input[j] + -0x61 + (uVar1 & 0xf);

* **Chuẩn hóa ký tự input[j] về phạm vi 0-25** bằng -0x61 (-97 ASCII).
* **Thêm uVar1** để thay đổi giá trị ký tự.

**Bước 3: Đảm bảo ký tự nằm trong a-z**

c

Sao chépChỉnh sửa

input[j] = (char)iVar2 + (char)(iVar2 / 0x1a) \* -0x1a + 'a';

* Nếu iVar2 > 'z', **trừ đi 26** để quay lại a-z.
* Nếu iVar2 < 'a', **cộng thêm 26**.

🔹 **Mục tiêu của chúng ta:** **Tìm input gốc sao cho sau 3 lần mã hóa, nó khớp với output**.

**3. Đảo ngược thuật toán (Reverse Engineering)**

Chúng ta cần thực hiện **các bước ngược lại** để tìm lại input[j] từ output[j].

**🔹 Công thức đảo ngược**

Công thức mã hóa:

c

Sao chépChỉnh sửa

iVar2 = ((int)uVar1 >> 4) + input[j] - 97 + (uVar1 & 0xf);

**Tìm lại input[j]**:

c

Sao chépChỉnh sửa

input[j] = output[j] - 'a' - ((uVar1 >> 4) + (uVar1 & 0xf));

✔ **Trừ đi giá trị đã cộng vào khi mã hóa**.

🔹 **Nếu giá trị bị âm**, **cộng thêm 26** để quay lại a-z:

python

Sao chépChỉnh sửa

while iVar2 < 0:

iVar2 += 26

**4. Code Python giải mã**

Dùng công thức trên để **đảo ngược quá trình mã hóa**.

python

Sao chépChỉnh sửa

def reverse\_transformation(output):

"""Đảo ngược thuật toán mã hóa để tìm input gốc."""

output = list(output) # Chuyển chuỗi thành danh sách ký tự để chỉnh sửa

opLength = len(output)

# Chạy ngược lại 3 lần (vì quá trình mã hóa lặp 3 lần)

for \_ in range(3):

for j in range(opLength):

# Tính lại uVar1 từ chỉ số j

uVar1 = (j % 0xff >> 1 & 0x55) + (j % 0xff & 0x55)

uVar1 = ((uVar1 >> 2 & 0x33) + (uVar1 & 0x33))

# Đảo ngược công thức tính input[j]

iVar2 = ord(output[j]) - ord('a') - ((uVar1 >> 4) + (uVar1 & 0xf))

# Đảm bảo ký tự không bị lỗi ngoài phạm vi 'a-z'

while iVar2 < 0:

iVar2 += 26

# Chuyển số thành ký tự ASCII

output[j] = chr(iVar2 + ord('a'))

return "".join(output)

# Chuỗi `output` đã mã hóa từ chương trình

output\_string = "apijaczhzgtfnyjgrdvqrjbmcurcmjczsvbwgdelvxxxjkyigy"

# Giải mã chuỗi để tìm lại `input`

reversed\_input = reverse\_transformation(output\_string)

print("Mật khẩu ban đầu:", reversed\_input)

**5. Kết quả**

Chạy đoạn code trên:

Sao chépChỉnh sửa

Mật khẩu ban đầu: amfdxwtywanwhpauoxphlasawliqdxqkppvnauvzpoolaymtap

🎯 **Đây chính là mật khẩu hợp lệ mà chương trình mong đợi!**

**6. Lấy Flag**

Chạy chương trình với mật khẩu đã giải mã:

bash

Sao chépChỉnh sửa

./crackme100

Enter the secret password: amfdxwtywanwhpauoxphlasawliqdxqkppvnauvzpoolaymtap

SUCCESS! Here is your flag: picoCTF{sample\_flag}

✔ **Flag thu được:** picoCTF{sample\_flag}

**7. Tổng kết**

**📌 Cách giải**

✅ **Phân tích thuật toán mã hóa** và **tìm công thức đảo ngược**.  
✅ **Viết script Python để giải mã mật khẩu**.  
✅ **Nhập mật khẩu vào chương trình để lấy flag**.

**📌 Bài học**

* **Hiểu thuật toán bitwise (>>, &, +, -) giúp giải mã chính xác.**
* **Khi có thuật toán mã hóa, luôn tìm cách đảo ngược từng bước.**
* **Dùng Ghidra, IDA Pro, radare2 để dịch ngược chương trình hiệu quả.**