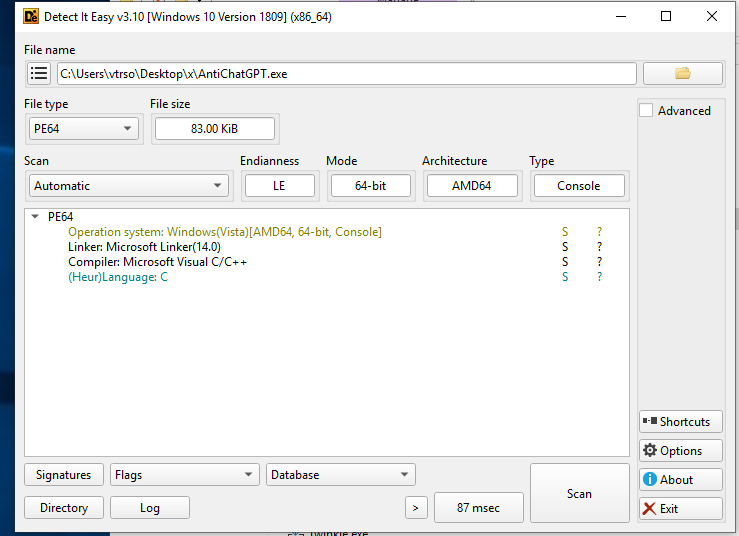
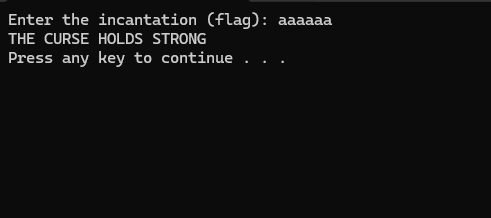
Write up: AntiChatGPT challenge

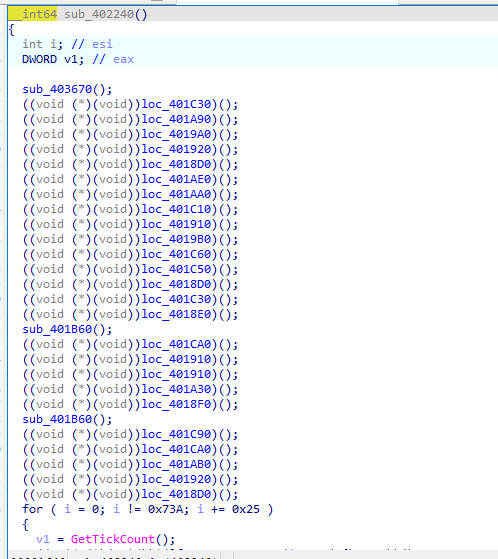
Ném chương trình vào Detect It Easy, ta có thể thấy chương trình 64 bit và được viết bằng c.



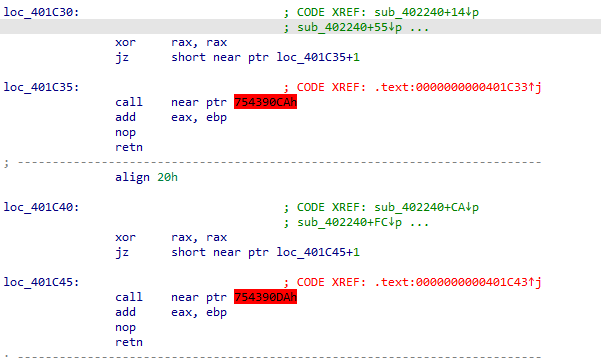


Chương trình chỉ đơn giản yêu cầu chúng ta nhập flag, nếu sai sẽ trả về “THE CURSE HOLDS STRONG”

Sử dụng ida64 để tiến hành phân tích. Bắt đầu tìm hàm main của challenge

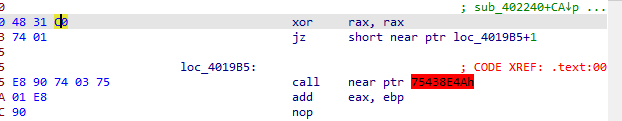


Chương trình đã bị làm rối bởi các mã rác vô nghĩa khiến cho việc tìm kiếm hàm chứa chức năng chính trở nên khó khăn.

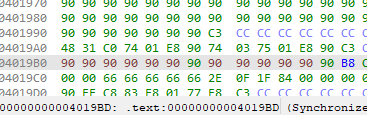


Khi ta thử kiểm tra vài hàm đầu tiên, tác giả đã sử dụng kỹ thuật anti disassembly để thể ngăn cản quá trình dịch ngược

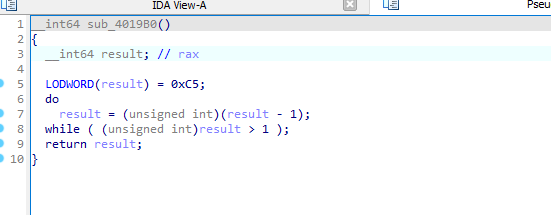
Ta có thể thấy tác giả đều dùng 1 dạng format giống nhau ở đầu mỗi hàm để ida không nhận diện được.



Ta có thể patch bằng tay hoặc viết script patch 1 cách dễ dàng để có thể disassembly bình thường



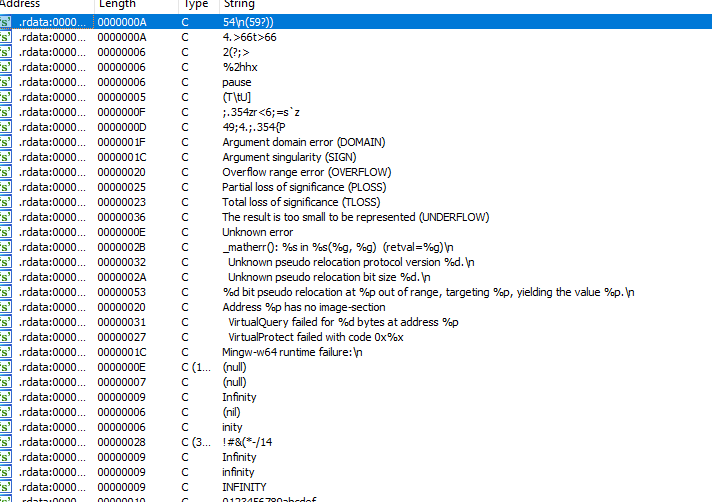
Sau khi thực hiện patch junk code ta có thể thấy hàm đã bình thương trở lại



Sau khi đã patch xong hết, điều bây giờ cần làm là tìm hàm chính so sánh flag.

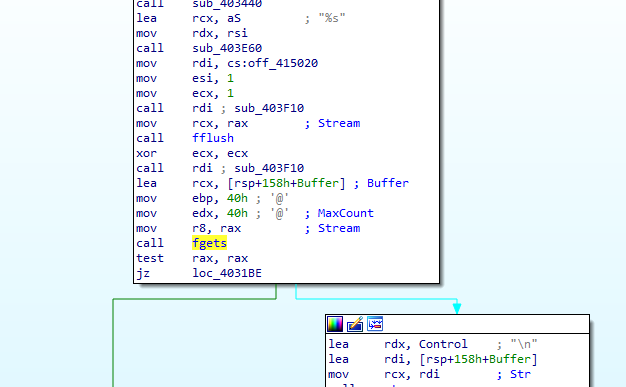
Khi ta chạy chương trình có thể thấy chương trình in ra các string, ta có thể dựa vào đó để lần ra.

Sử dụng shift+f12 để xem string



Có vẻ các string đã được mã hóa

Vậy ta sẽ đổi hướng sang tìm các hàm như printf, fgets



Vậy là ta đã tìm ra điểm đầu vào

Chương trình này mã hóa flag theo dạng block cipher tự chế (giống Feistel network) với:

· Key bí mật: "Th1s\_1s\_A\_V3ry\_S3cr3t\_K3y\_F0r\_CTF!"

· Kích thước block: 8 byte (64 bit)

· Số round: 8 vòng mã hóa

· Bảng tra cứu (S-box): byte\_4120F0 dùng trong F\_function

**Các bước thực hiện mã hóa**

Flag gốc → chia thành các block 8 byte →

Sinh round keys từ key bí mật →

Mã hóa từng block → Kết quả là ciphertext

**Hàm sinh round key (generate\_round\_keys)**

Hàm này nhận key bí mật (được coi là mảng các int) và tạo ra 8 round keys.

Nguyên tắc:

Dùng phép xoay bit trái (ROL), cộng/trừ, XOR với hằng số.

Kết quả cuối cùng là mảng a2[0..7] (8 khóa cho 8 vòng).

**Hàm encrypt**

Hàm này nhận:

-Một block dữ liệu 64-bit (a1[0] và a1[1] mỗi phần là 32-bit).

-Mảng round keys (a2).

Quy trình (giống Feistel cipher):

1.Lấy nửa trái L và nửa phải R của block.

2.Lặp qua 8 round:

* R\_new = L ^ F(R, round\_key[i])
* Đổi chỗ L ↔ R.

3.Sau vòng cuối, dữ liệu đã mã hóa.

**Hàm F\_function**

Đây là hàm trộn bit:

1.XOR giá trị đầu vào với round key.

2.Tách thành từng byte, tra trong bảng byte\_4120F0 (S-box).

3.Ghép lại, xoay trái 13 bit (ROL4(..., 0xD)).

4.Trừ đi hằng số 0x61C88647.

**Mã hóa toàn bộ flag**

Chương trình chạy vòng do { encrypt(...); } while(...) để mã hóa từng block 8 byte của flag.

Output cuối là flag đã mã hóa (ciphertext).

**Cách giải mã**

Vì đây là Feistel cipher, giải mã dùng cùng round keys nhưng chạy ngược vòng:

Nếu mã hóa chạy round 0 → round 7 thì giải mã chạy round 7 → round 0.

Công thức đảo ngược:

*L\_old = R\_new ^ F(L\_new, round\_key[i])*

*R\_old = L\_new*

**Các bước mình thực hiện:**

1.Trích xuất ciphertext từ file.

2.Sinh round keys từ key bí mật bằng generate\_round\_keys.

3.Viết hàm decrypt đảo ngược encrypt:

4.Dùng Feistel đảo ngược, chạy từ round cuối về round đầu.

5.Giải mã từng block 8 byte → ghép lại thành plaintext.

Kết quả:

PTITCTF{k1ng\_0f\_Pt1t\_NigM4o\_z3ro\_d4Y\_zxo}