**Challenge: canary**

- Canary là một giá trị ngẫu nhiên được đặt giữa buffer và return address trên stack. Mục tiêu là để phát hiện ghi đè tràn bộ đệm:

- Trước khi hàm trả về, chương trình sẽ kiểm tra canary. Nếu bị thay đổi -> chương trình kết thúc ngay.

- Các chế độ bảo vệ ảnh hưởng đến việc bypass canary:



- Với file chall:

A screen shot of a computer program

AI-generated content may be incorrect.

- Canary found -> cần leak

- NX enabled -> không dùng đc shellcode, nhưng có thể dùng ROP để gọi system(“/bin/sh”)

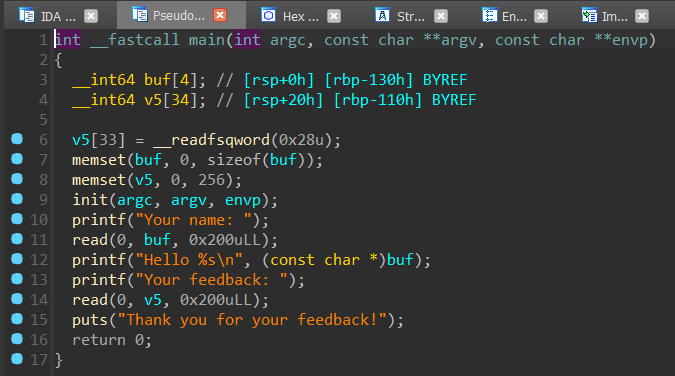
- No PIE -> biết đc địa chỉ hàm, dễ tìm gadget và gọi thẳng vào win() hoặc system() nếu có

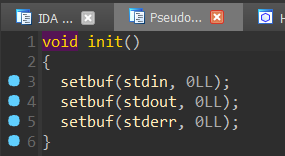
- Stripped: No -> dễ reverse, dễ phân tích

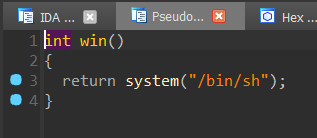
**\* Xem pseudo code ở IDA xem, chương trình hoạt động ntn:**

- Có 3 hàm chính: main(), init(), win()

Mô tả chương trình hoạt động: Khai báo 2 chuỗi buf với v5, gán tất cả các phần tử trong 2 chuỗi giá trị thành giá trị null sau đó khai báo hàm init khởi tạo các bộ điệm, nhập lần lượt 2 chuỗi buf và v5 rồi kết thúc chương trình







-> Mục tiêu: ta thấy khi nhập đúng 0x200 byte hết khả năng của chuỗi sẽ thấy chương trình sẽ bị exit luôn do canary. Như vậy, cách duy nhất ta có thể làm là leak canary ra nếu có buffer overflow

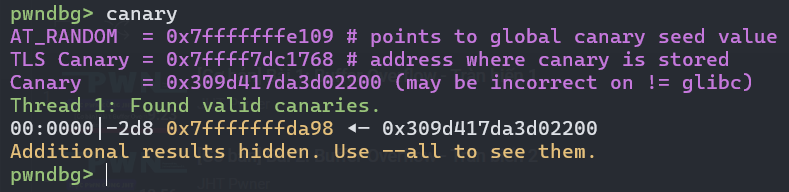
**\* Tiến hành debug thử chương trình:**

- Ta thấy cái fs[0x28] là vị trí mà chứa cái giá trị canary, thì giá trị của canary nó chính là raxA screenshot of a computer

AI-generated content may be incorrect.

- Đặc điểm của canary: byte đầu tiên luôn là byte null còn 7 byte còn lại sẽ là ngẫu nhiên và khác null, giá trị canary sẽ khác nhau sau mỗi lần chạy

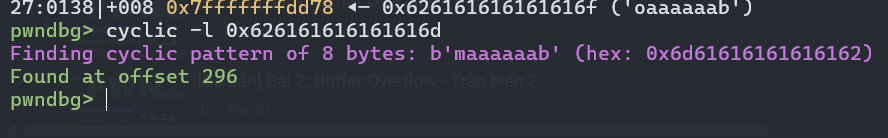
- Ta có thể dùng lệnh tel để tìm giá trị của canary, thường nó sẽ nằm ở trước saved sbp, saved rip, ta có thể kiểm tra bằng canary nó sẽ cho đúng giá trị canary



- Cách leak: ta biết rằng hàm read sẽ không tự động thêm null byte sau khi nhập do có ta có thể cho nó nối với cái canary, ta sẽ nhập tràn xuống đúng cái canary, mình sẽ overwrite 1 byte null của canary, sau nó tận dụng hàm printf ngay sau để in ra 1 dãy mình nhập vào cộng với 7 byte canary tìm được.

**\* Stage 1: Viết script leak canary**

- Đầu tiên ta sẽ tìm offset từ lúc nhập buf cho tới canary (dùng cyclic kết hợp với tel)



-> offset = 296 + 1 ( Do ta phải overwrite của 1 byte null của canary vì hàm printf sẽ ngắt nếu gặp null, sau khi overwrite đc byte null thì printf sẽ ko thấy byte null sẽ in tiếp 7 byte của canary)

- Ta có script như sau:

A computer code with numbers and symbols

AI-generated content may be incorrect.

Kiểm tra lại bằng debug động xem đúng chưa:

A computer code with numbers and letters

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

-> chính xác

**\* Stage 2: Viết script lấy shell**

- Ta sẽ viết 1 cái payload giữ nguyên cái giá trị canary vừa leak đó và sau đó overwrite ret thành địa chỉ hàm main là xong (ret2win) do có sẵn hàm win trong chương trình.

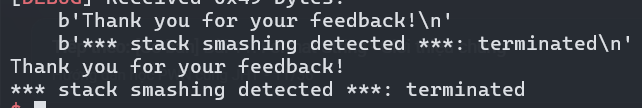
- Đầu tiên vẫn truyền vào 296 byte rác rồi là giá trị canary (để bypass canary) sau đó đến rbp và cuối cùng sẽ điều khiển đc rip thôi)

- Script lấy shell:

A computer code with text

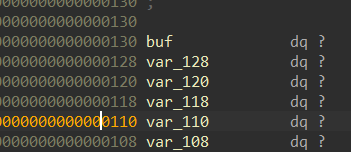
AI-generated content may be incorrect.

- Kiểm tra bằng debug động:



-> Thấy canary bị sai, khả năng do truyền thừa hoặc thiếu byte rác, ta sẽ tính lại offset từ lúc nhập v5 đến ret.

- Do v5 khai báo sau buf khoảng 0x20 byte, nên ta cần trừ bớt 0x20 byte vào truyền đúng canary



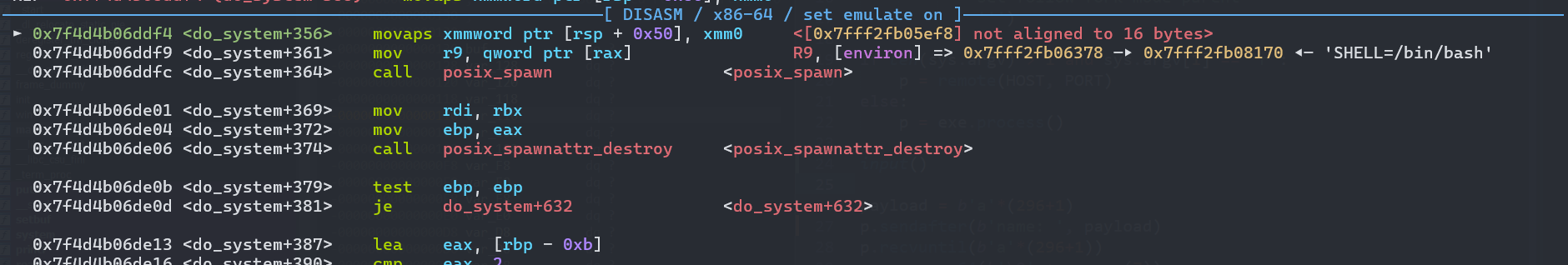
-> offset = 296 – 0x20

Ta có script mới :

A computer code with text

AI-generated content may be incorrect.

- Tiếp tục kiểm tra bằng debug động:

  
- Ta thấy lỗi 0x7fff2fb05ef8 ko chia hết cho 16. Ta có 2 hướng xử lý có thể nhảy trực tiếp vào win (+5 byte) hoặc tìm 1 địa chỉ của hàm ret nào đó trước khi truyền địa chỉ hàm win.

- Script cuối:

A computer code with text

AI-generated content may be incorrect.

- Kiểm tra xem oke chưa:  
A screenshot of a computer

AI-generated content may be incorrect.

-> Đã chiếm đc shell