PLT,GOT 추가 자료

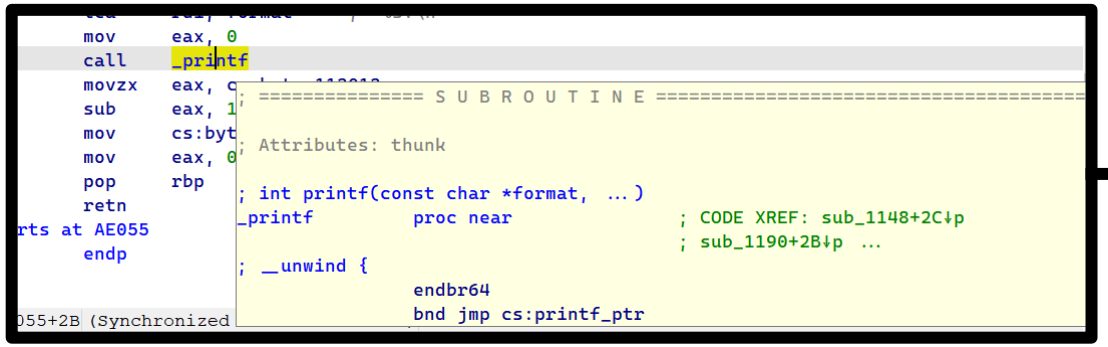
유효곤

PLT와 GOT가 무엇인가?

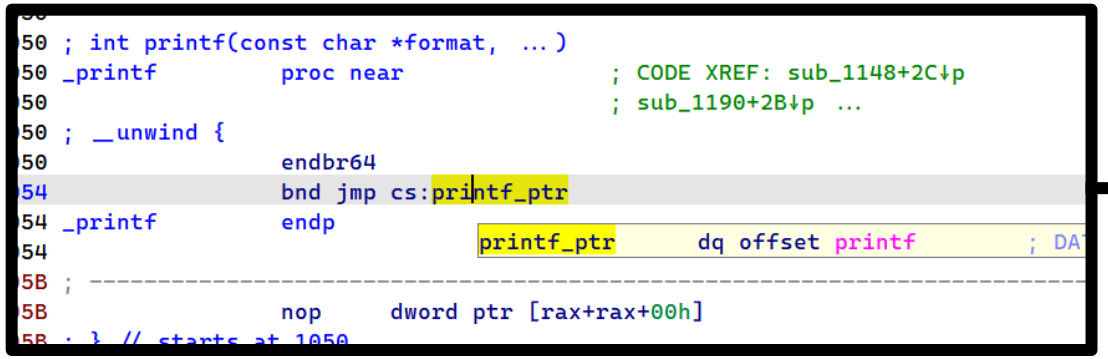
- PLT : Procedure Linkage Table
  - exec권한이 있는 부분으로, GOT값을 참조해서, 라이브러리의 실제 함수의 entry point로 jmp함.
  - **GOT를 처음 참조할** 때는 "_dl_runtime_resolve"라는 함수가 실행되어 라이브러리의 함수의 주소를 GOT에 저장하고, 이후 이 값을 참조하여 실제함수를 실행함.

- GOT : Global Offset Table
  - 라이브러리의 함수/변수의 주소를 저장함.

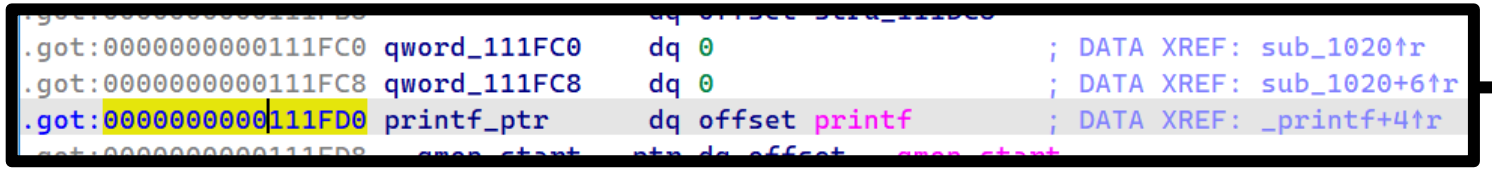
printf를 호출하는 과정으로 알아보자



printf를 호출하는 부분
call _printf 명령어를 실행함

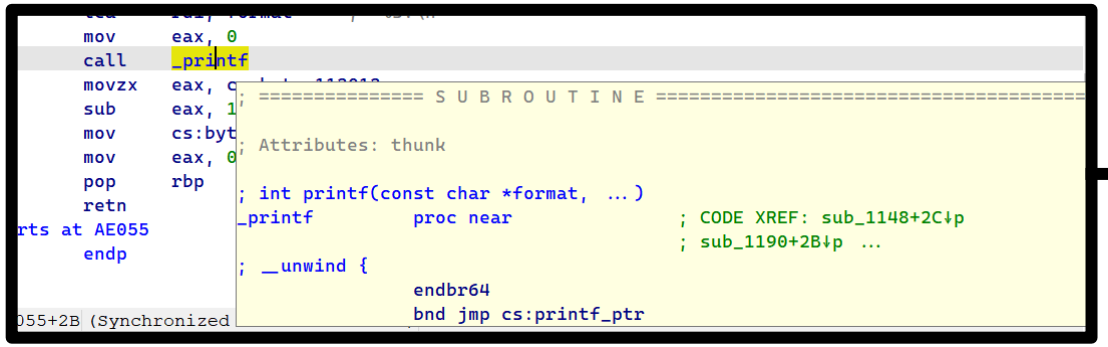


_printf를 따라가서 확인해본 부분
실제로, printf가 있는 것이 아니라,
jmp cs:printf_ptr를 실행하는 것을 확인할 수 있음.



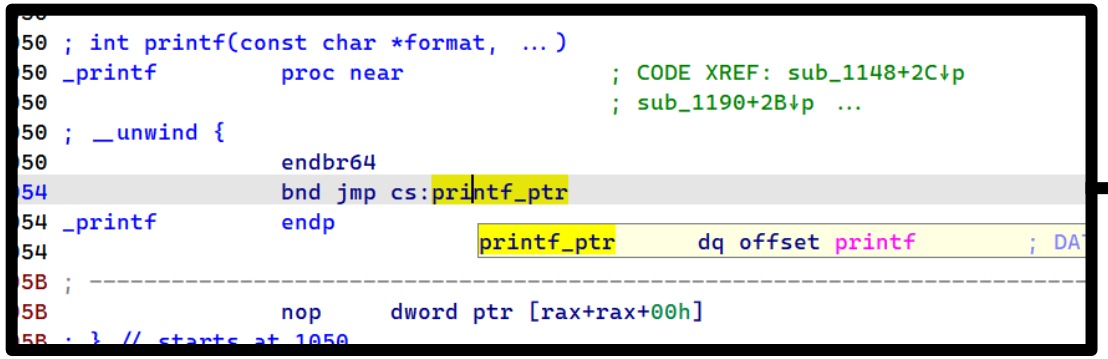
printf_ptr의 위치를 가보면 printf의
offset을 저장하고 있다고 나옴.

printf를 호출하는 과정으로 알아보자



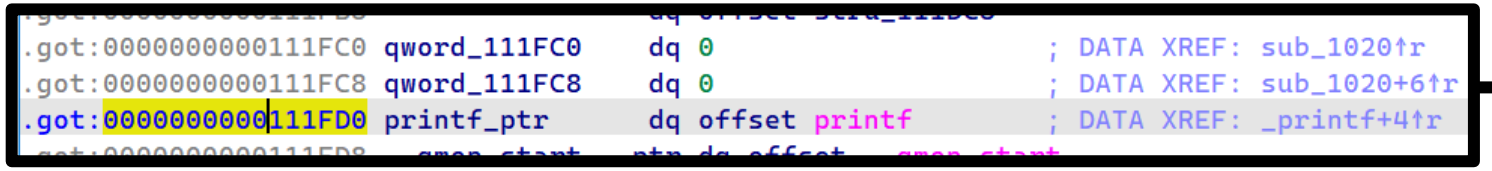
함수 호출

printf를 호출하는 부분
call _printf 명령어를 실행함



PLT

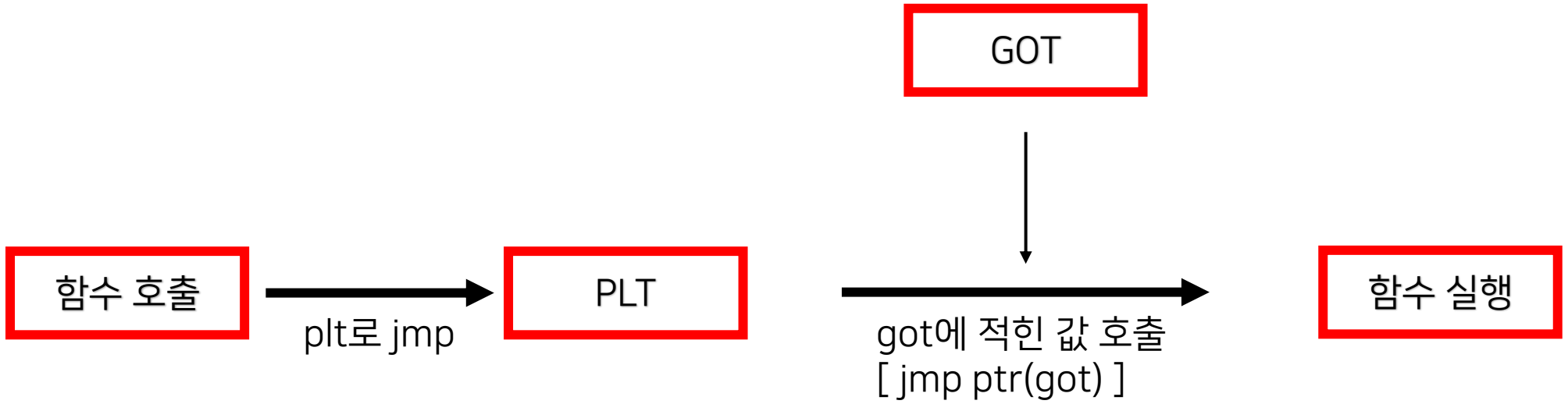
_printf를 따라가서 확인해본 부분
실제로, printf가 있는 것이 아니라,
jmp cs:printf_ptr를 실행하는 것을 확인할 수 있음.



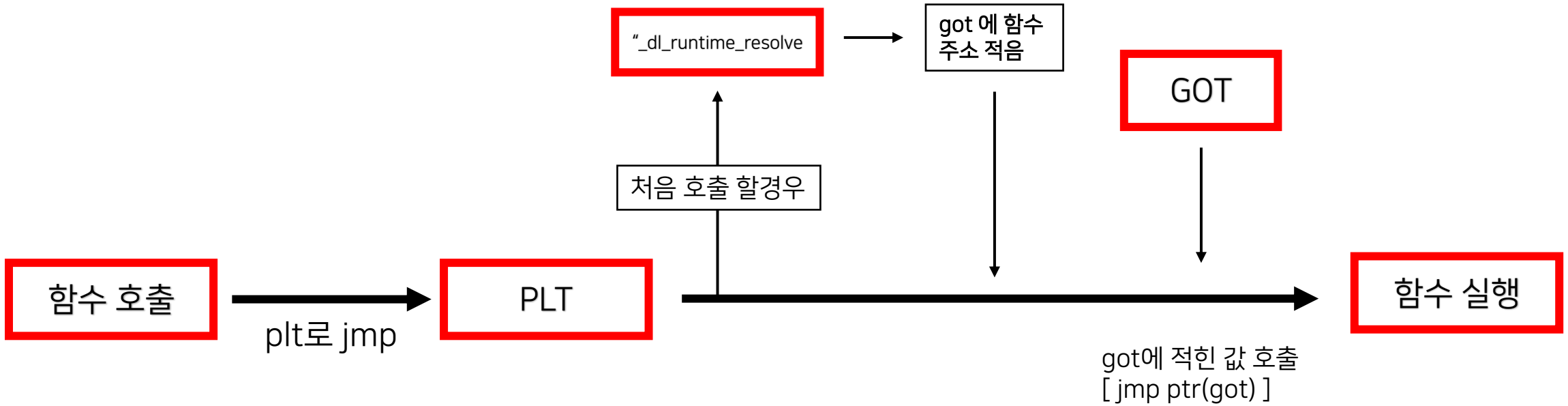
GOT

printf_ptr의 위치를 가보면 printf의
offset을 저장하고 있다고 나옴.

즉, 함수는 이렇게 호출된다.
(got에 함수 주소가 적혀있을 경우 = 처음 호출이 아닌 경우)



만약 처음 함수를 호출하는 경우?
(got에 값이 적혀있지 않을 경우)



왜 PLT, GOT를 사용하는 가?

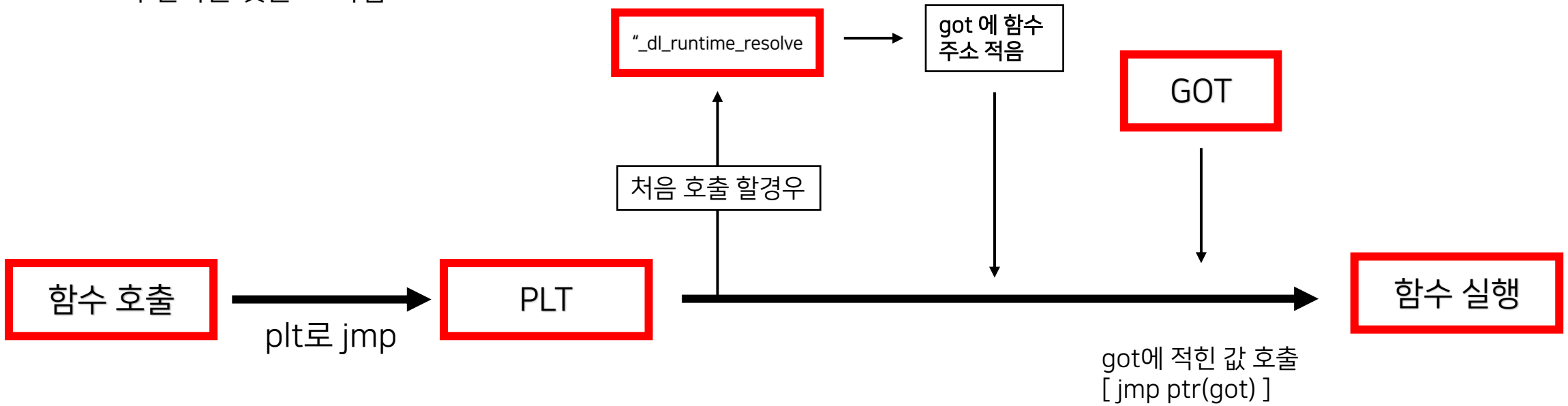
- 용량의 압축을 위해서!
  - 바이너리에 함수를 포함하게 되면 용량이 어마어마해짐 ㅇㅇ
  - 함수를 포함한 걸 보고 static linking이라고 함.

보호 기법

- PLT, GOT와 관련있는 보호기법이 RELRO임.
  - 정확히는 Partial Relro와 Full Relro에서 got부분을 w권한을 주느냐 마느냐의 차이긴 한데, 이건 다음주에 설명 해줄 거임.
  - 다음페이지에 설명할 GOT Overwrite는 Partial Relro의 상황에서만 사용 가능

GOT Overwrite

- 함수를 실행하는 과정은 아래와 같음.
- 그런데, GOT에 주소를 적는 다는 것은 GOT에 RW권한이 둘다 있는 것을 말함.
- 그래서 우리가 GOT값을 잘 조작할 수 있음.
- 그러면, 우리가 printf의 GOT에다가 SYSTEM의 offset을 쓰게 되면?
  - Code 에서 printf("ls")을 하려던게, 실제로는 system("ls")를 실행하는 것과 같아지게 됨.
- 그래서 직접, Ret Addr을 system으로 옮기지 않아도, GOT값을 조절하는 것으로도 System 실행이 가능함.
  - 사실 got값에다가 원하는 주소를 적어 넣으면 그 주소에 적힌 code를 실행하게 됨. 그래서 꼭 system의 주소를 넣어야 한다는 것은 또 아님



과제 화이팅~!
모르는 거 생기면 물어보세요~