Section 09

리턴 어드레스

혹시 눈치 채신 분 계신가요?

바로 이 공간에 "리턴 어드레스"가 위치하고 있는 것입니다.
두 개의 공간 중 오른쪽에 리턴 어드레스가 위치하고 있습니다.

[전체모습]

리턴 어드레스!

0xbfffee54		0xbfffee5c	0xbfffee60	0xbfffee64	0xbfffee7c	0xbfffee80	0xbfffee84
4바이트		4바이트	4바이트	4바이트	4바이트	4바이트	4바이트
200		다음 코드의 주소	10	20	200	20	10

이는 초창기 프로그래머들이 리턴 어드레스를 이 곳에 위치시켰을 때 가장 효율적일 거라고 판단을 했기 때문일 것입니다. 그리고 실제 어셈블리어 레벨에서 함수가 호출되고 복귀되는 과정을 분석해 보면(심화편), 왜 그렇게 설계를 했는지를 어느정도 추측할 수 있습니다.

그리고 왼쪽엔 SFP(Saved Frame Pointer)라는 값이 저장되어 있는데, 일단 얘는 무시하셔도 됩니다.

[전체모습]

SFP 리턴 어드레스!

0xbfffee54	0xbfffee58	0xbfffee5c	0xbfffee60	0xbfffee64	0xbfffee7c	0xbfffee80	0xbfffee84
4바이트	4바이트	4바이트	4바이트	4바이트	4바이트	4바이트	4바이트
200	sfp값	다음 코드의 주소	10	20	200	20	10

9.리턴 어드레스

리턴 어드레스가 저장되는 메모리 주소엔 항상 고정된 값이 들어가 있는 것이 아니라, 함수가 호출될 때마다 그에 적합한 값, 즉 다음에 실행될 코드의 주소가 저장되게 됩니다.

이렇게 해서 한 함수가 다른 함수를 호출할 때의 메모리 구조를 모두 그려보았는데요.

자! 여기서 우리는 무시무시한 사실을 발견할 수 있습니다.
이러한 메모리 구조로 볼 때, 만약 호출된 함수의 지역 변수를 시작으로
버퍼 오버플로우가 발생하는 취약점이 존재한다면,
리턴 어드레스를 덮어씌울 수 있는 구조로 되어 있다는 점입니다.

지난 무임승차 문제의 예에서 auth 값을 덮어씌운 것과 동일한 방식으로 리턴 어드레스를 덮어 씌울 수 있는 것입니다.

[전체모습]

0xbfffee54	0xbfffee58	0xbfffee5c	0xbfffee60	0xbfffee64	0xbfffee7c	0xbfffee80	0xbfffee84
4바이트	4바이트	4바이트	4바이트	4바이트	4바이트	4바이트	4바이트
200	sfp값	다음 코드의 주소	10	20	200	20	10

리턴 어드레스는 다음에 실행 될 코드의 주소를 담고 있다고 그랬는데, 그 값이 변조 가능하다면!?

다음에 실행 될 코드의 주소를 우리 마음대로 선택하는 것이
가능하단 말이 됩니다!


대학교도 내 마음대로
선택할 수 있었으면
참으로 좋겠다!
기왕이면 여자친구도 말이야!