**스택 프레임 구조 요약 설명**

| **{Buffer}** |
| --- |
| {변수} |
| {RBP(Base pointer)} |
| {RET(Return address)} |

* RET는 리턴 어드레스를 뜻하고, 현재 실행되고 있는 함수에서 다음에 어디로 돌아가야 하는지를 가리키는 주소.(함수가 실행을 마치고 돌아가야할 위치를 가리키는것)(취약점 활용을 할 수 있다)

함수 호출이 끝난 후, 제어를 원래 호출한 위치로 돌려주는 데 사용됩니다. 함수가 호출될 때 현재 명령어의 주소를 RET에 저장하고, 함수가 끝나면 RET에 저장된 주소로 복귀합니다. 이는 함수가 여러 번 호출되어도 정확히 호출된 위치로 돌아가도록 보장합니다. RET는 스택에 저장되어 함수 호출 순서를 유지하며, 호출과 복귀의 일관성을 유지하는 핵심 요소입니다.

* RBP는 베이스 포인터를 뜻하고, 스택 프레임의 시작을 알린다.(스택 프레임의 시작점)

스택 프레임의 기준점 역할을 합니다. 함수가 호출될 때 현재 RBP는 스택에 저장되고, 새 스택 프레임의 시작 주소로 설정됩니다. 이를 통해 함수 내의 지역 변수와 매개변수를 상대적으로 쉽게 참조할 수 있습니다. 함수가 끝날 때는 저장된 RBP 값을 복원하여 이전 스택 프레임으로 돌아갑니다. RBP는 스택의 일관성을 유지하고, 함수 간의 메모리 참조를 명확히 하는 데 중요한 역할을 합니다.

**스택 프레임의 작동 원리**

1. 함수 호출 시

* 함수가 호출되면 현재 명령어 주소가 RET에 저장됩니다. 이는 함수 실행이 끝난 후 되돌아갈 주소입니다.
* 현재 RBP 값이 스택에 저장됩니다. 이를 통해 이전 스택 프레임으로 돌아갈 수 있습니다.
* RBP는 새로운 스택 프레임의 시작 주소로 설정됩니다.
* 함수의 매개변수와 지역 변수는 Buffer 영역에 저장됩니다.

1. 함수 실행 중

* 함수는 RBP를 기준으로 변수와 Buffer를 참조합니다. 이를 통해 지역 변수와 매개 변수를 쉽게 접근할 수 있습니다.

1. 함수 반환 시:

* RET에 저장된 주소로 제어가 이동합니다. 이는 호출한 위치로 돌아가도록 보장합니다.
* 스택에 저장된 이전 RBP 값이 복원됩니다. 이를 통해 이전 스택 프레임으로 돌아갑니다.
* 현재 스택 프레임이 제거되고, 이전 스택 프레임으로 돌아갑니다.

**스택프레임 취약점 정리**

**1. 버퍼 오버플로우(Buffer Overflow)**

버퍼 오버플로우는 프로그램이 고정된 길이의 버퍼보다 더 많은 데이터를 쓰려고 할 때 발생한다. 이로 인해 스택 프레임의 경계를 넘어 다른 데이터를 덮어쓸 수 있다. 공격자는 이를 악용해 다음과 같은 일을 할 수 있다

* RET(Return Address)를 덮어써서 악성 코드를 실행하도록 만들 수 있다.
* RBP(Base Pointer)를 덮어써서 프로그램 흐름을 변경할 수 있다. 버퍼 오버플로우는 프로그램의 예기치 않은 동작이나 시스템의 완전한 제어를 허용할 수 있는 취약점이다.

**2. 스택 스매싱(Stack Smashing)**

스택 스매싱은 버퍼 오버플로우의 한 형태로, 공격자가 스택프레임의 **RET** 값을 덮어써서 임의의 코드를 실행하도록 만드는 공격 기법이다. 공격자는 프로그램의 제어를 탈취하고 악성 코드를 실행할 수 있다.

**3. 포인터 변조(Pointer Manipulation)**

공격자는 버퍼 오버플로우를 통해 스택프레임에 저장된 포인터 변수를 조작할 수 있다. 예를 들어, 함수 포인터를 조작해 원래 의도한 함수 대신 공격자가 지정한 함수를 호출하게 만들 수 있다.

**4. 스택 오버플로우(Stack Overflow)**

재귀 함수가 종료 조건 없이 계속 호출되거나, 너무 많은 함수가 호출되어 스택 공간이 초과되면 스택 오버플로우가 발생할 수 있다. 이는 프로그램 크래시를 초래하고, 잠재적으로 시스템에 대한 공격 기회를 제공할 수 있다.