바이너리 실행파일의 버퍼 오버런 분석기

최재승
서울대학교 프로그래밍 연구실

2015.1.27 @ 12th Roasec Workshop

목표

- 바이너리 실행파일(binary executable)을 정적 분석하여 **버퍼 오버런** 취약점 탐지하기

- 소스 코드 없이 바이너리 파일로만 접근 가능한 소프트웨어도 분석하여 오류를 잡을 수 있도록

문제

- 바이너리에서 버퍼 오버런의 정의가 명확하지 않음
  - 프로그램의 메모리 구조와 관련된 정보가 명시적으로 드러나지 않음
  - e.g. 선언된 배열의 크기, 배열 접근 시 첨자(index)에 해당하는 값

(C source)

```
void foo(void)
{   int arr [32];
    int i ;
    for (i = 0; i <= 32; i++)
        arr [ i ] = 1; // overflow
}
```

(Compiled binary)

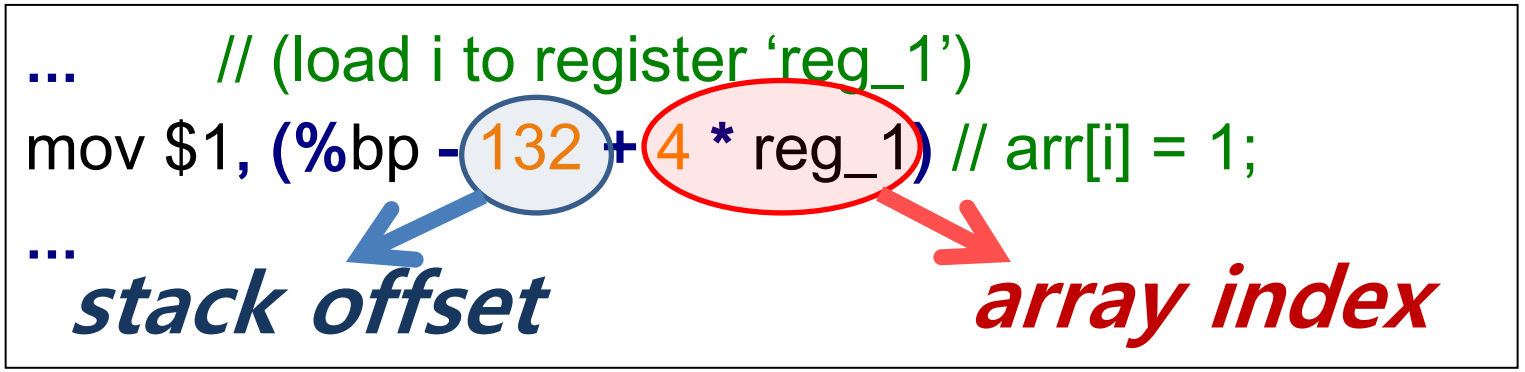
```
...
mov $0, (%bp - 4); // int i = 0;
cmp (%bp - 4), 32; // if (i > 32)
jg  label_end;      //   goto label_end;
mov (%bp - 4), %reg_1 // (load i to register)
mov $1, (%bp - 132 + 4 * reg_1) // arr[i] = 1;
...
```

관찰

- 간단한 휴리스틱으로, **스택 오프셋**에 해당하는 값과 **배열 첨자**에 해당하는 값을 구분 가능

  *스택 오프셋 : 스택 프레임 내에서 지역변수 위치를 정해주는 정수 값

- 스탯 오프셋 값 : 어셈블리에 상수로 박혀(hard-coded) 있음
- 배열 인덱스 값 : 레지스터나 메모리로부터 읽어옴

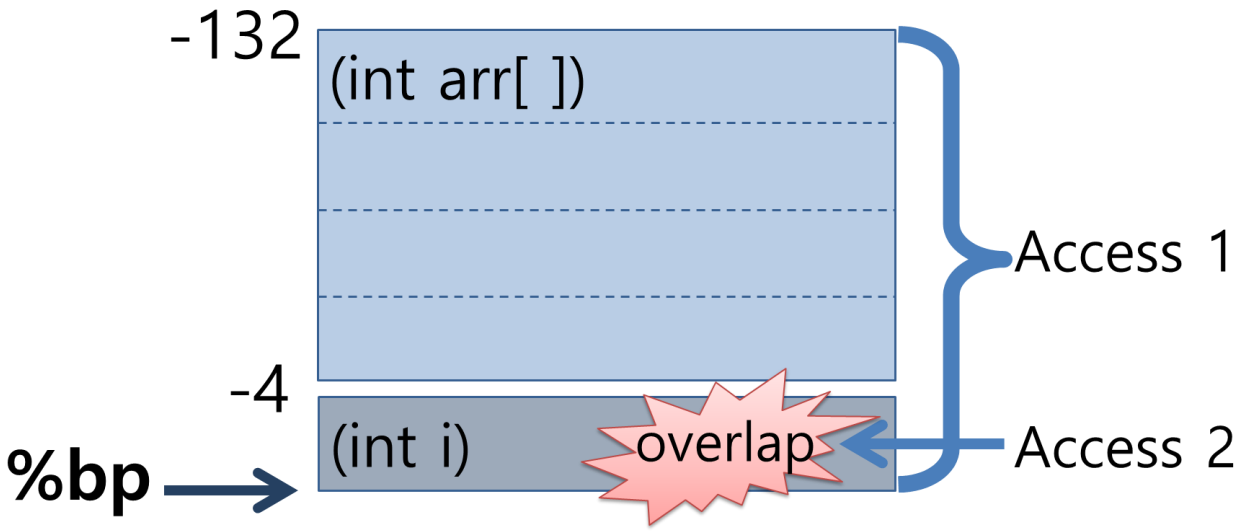


버퍼 오버런 (in binary)

- 두 메모리 주소의 스택 오프셋 값이 다르면, 서로 다른 지역 변수를 가리키는 것으로 가정
- 그러한 두 주소가 접근하는 메모리 공간이 겹쳐진다면 버퍼 오버런으로 판단 가능

```
mov $1, (%bp - 132 + 4 * reg_1) // arr [x] = 1;

mov $2, (%bp - 4) // y = 2
```



분석기

- 요약 해석 틀에 기반하여 분석기를 설계
  - 이러한 관찰과 직관을 반영한 요약 실행 의미 정의
- 프로토타입 구현 및 간단한 예제로 실험 중
- 포스터 세션에서 더욱 자세한 이야기를 준비하겠습니다

감사합니다