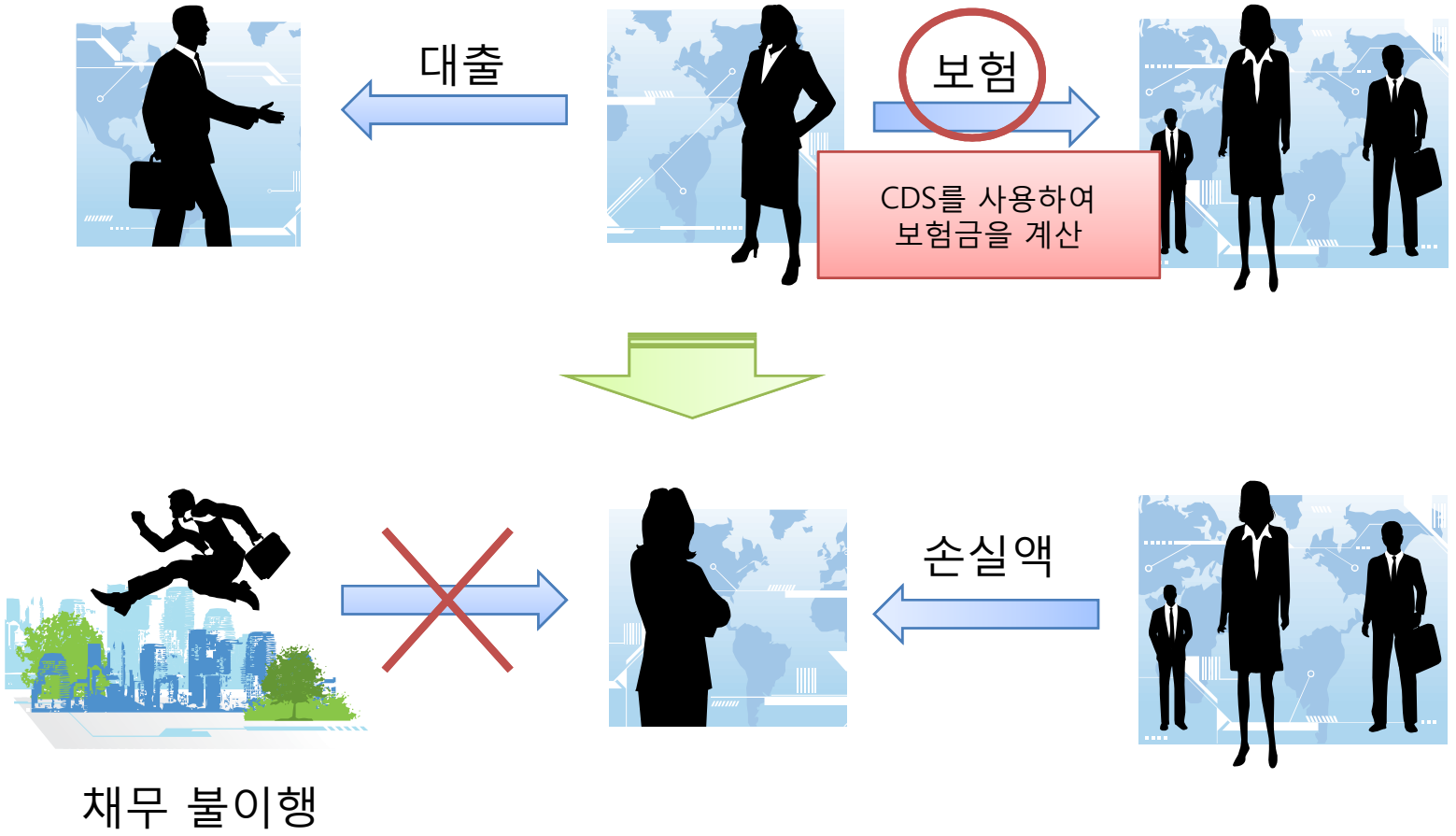
Specializing Airac in J.P. Morgan CDS Software

고윤석
소프트웨어 무결점 연구센터
서울대학교
2009. 7. 10.

J.P. Morgan CDS Software?

⊙ 신용 파산 스왑(CDS: Credit Default Swap)



J.P. Morgan CDS Software?

CDS 시장의 규모

(작년기준) 약 60조 달러

J.P. Morgan CDS Software?

소스코드 크기

- 약 20,000라인(전처리 이전, 헤더를 제외한 코드 기준)

Airac?

일반적인 C 프로그램에 대해서 버퍼오버런을 검증하는 분석기

- 분석이 부정확할 수 있음
- 안전하게 검증

목표

Airac을 CDS Software에 특화시키자

- 허위경보가 하나도 없도록

초기 분석 결과

Airac으로 CDS Software를 검증한 결과

- 경보 631개
- 대부분의 경보가 허위 경보일 것으로 추정

허위경보 제거 – 경우 1

- **원인**
  - 함수의 결과값이 매우 부정확함

허위경보 제거 – 경우 1

⊙ **원인**

- 함수의 결과값이 매우 부정확함

⊙ **이유**

- 함수의 시작점에서 인수 값을 모두 합침

예

```
void foo()
{
    char* array = (char*)JpmcdsMallocSafe(10);
    for (int i=0; i<10; i++)
        array[i] = 10L;
}
```

array = Ablk([10,10])

array = Ablk([10,20])

```
void bar()
{
    char* array = (char*)JpmcdsMallocSafe(20);
    for (int i=0; i<20; i++)
        array[i] = 20L;
}
```

허위경보 발생!

array = Ablk([10,10])

array = Ablk([10,20])

```
void* JpmcdsMallocSafe(size_t size)
{
    void *ptr = malloc(size);

    …   // error 처리

    return ptr;
}
```

size = [10, 10]

size = [10, 20]

ptr = Ablk([10,10])

ptr = Ablk([10,20])

허위경보 제거 – 경우 1

⊙ 원인

- 함수의 결과값이 매우 부정확함

⊙ 이유

- 함수의 시작점에서 인수 값을 모두 합침

⊙ 해결방안

- 함수 펼치기 - 함수 호출위치에 함수의 몸체를 그대로 넣음
- CDS Software는 전체 소스의 양이 작기 때문에 함수 펼치기가 가능

예

```
void foo()
{
    char* array = (char*)JpmcdsMallocSafe(10);
    for (int i=0; i<10; i++)
        array[i] = 10L;
}
```

```
void* JpmcdsMallocSafe(size_t size)
{
    void *ptr = malloc(size);

    …   // error 처리

    return ptr;
}
```

```
void foo()
{
    size_t size = 10;
    void* ptr = malloc(size)

    … // error 처리

    char* array = (char*)ptr
    for (int i=0; i<10; i++)
        array[i] = 10L;
}
```

허위경보 제거 – 경우 1

⊙ 원인

- 함수의 결과값이 매우 부정확함

⊙ 이유

- 함수의 시작점에서 인수 값을 모두 합침

⊙ 해결방안

- 함수 펼치기 - 함수 호출위치에 함수의 몸체를 그대로 넣음
- CDS Software는 전체 소스의 양이 작기 때문에 함수 펼치기가 가능

⊙ 함수 펼치기 대상은?

- 프로그램 내에서 자주 호출되는 함수들을 찾아냄

허위경보 제거 – 경우 1

정적 호출 그래프(Static Call Graph) 분석 결과:

전체 선언된 함수 - 297개

9번 이상 호출되는 함수 - **14개**

허위경보 제거 – 경우 1

함수 펼치기 적용 결과

- 전체 명령 수 40% 증가
- 허위경보 461개 제거
- 분석 속도 47%로 감소

허위경보 제거 – 경우 2

⊙ 원인

- strlen함수 호출 결과가 매우 부정확함

허위경보 제거 – 경우 2

⊙ 원인

- strlen함수 호출 결과가 매우 부정확함

⊙ 이유

- Airac은 배열의 값을 하나로 뭉쳐서 관리함

예

```
TCurve* BuildExampleZeroCurve()
{
    char *types = "MMMMMSSSSSSSSS";
    …
    int n = strlen(types);


    TDate* dates = JpmcdsMallocSafe(sizeof(TDate)*n);
    …
}
```

types = Ablk([14,14])

types[i] = [0, 83]

n = [0,]

dates = Ablk([0,])

허위경보 제거 – 경우 2

⊙ 원인

- strlen함수 호출 결과가 매우 부정확함

⊙ 이유

- Airac은 배열의 값을 하나로 뭉쳐서 관리함

⊙ 해결방안

- 배열 도메인을 확장하여 0이 할당된 위치를 저장
- 표준함수 strlen의 의미를 만들어 넣음

예

```
TCurve* BuildExampleZeroCurve()
{
    char *types = "MMMMMSSSSSSSSS";
    …
    int n = strlen(types);


    TDate* dates = JpmcdsMallocSafe(sizeof(TDate)*n);
    …
}
```

types = Ablk([14,14], [13,13])

n = [13, 13]

dates = Ablk([13, 13])

허위경보 제거 – 경우 2

⊙ 배열도메인 확장 결과

- 허위 경보 14개 제거
- 버퍼의 크기를 알지 못하던 경보 중 20개는 버퍼의 크기를 알 수 있게 됨

결과

초기 전체 경보의 수		631 개
거짓 경보 줄이기	함수 펼치기	461 개
	배열 도메인 확장	14 개
	루프 시작점 조이기	1 개
	Airac 버그 수정	11 개
전체 줄어든 경보의 수		487 개
남은 경보의 수		144 개

- 분석시간이 155분에서 76분으로 단축

결론

- **분석기를 한 가지 소프트웨어에 특화시킬 수 있음**
  - 해당 소프트웨어의 특징에 맞게,
  - 부정확하게 분석하는 부분을 찾아서 그 부분을 정확하게 할 수 있도록,
  - 여전히 안전함은 유지할 수 있도록,

  - 631개의 경보에 대해서
    631가지의 방법을 적용할 필요는 없음

감사합니다.