DETECT INVALID MEMORY ACCESSES

T1.accessPredictor

> "T1.accessPredictor 덕분에
> 유효하지 않은 메모리 접근을 찾는데 소요되는 시간을
> 획기적으로 줄일 수 있었습니다. T1.access Predictor는 타겟에서
> 코드를 실행할 필요 없이 PC에서 바이너리를 분석하여
> 정적으로 잘못된 메모리 접근을 찾습니다."
>
> –T1.accessPredictor 사용 고객의 피드백–

CHECK MEMORY ACCESSES BASED ON THE BINARY ONLY

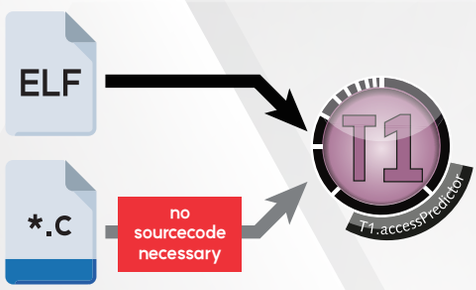
T1.accessPredictor를 사용하면 소프트웨어가 타겟에서 동작하기 전에 잘못된 메모리 접근을 감지할 수 있습니다.

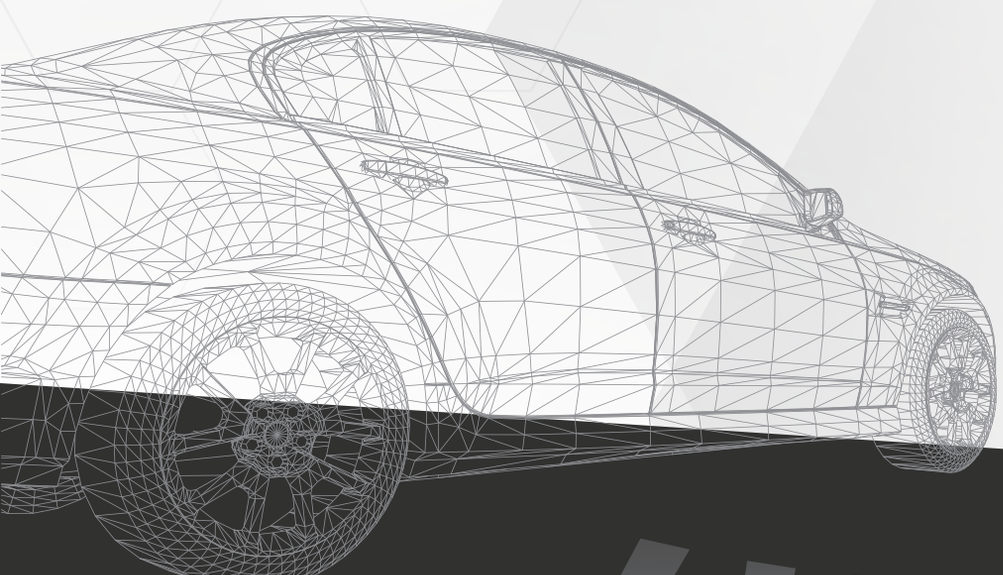
MPU(Memory Protection Unit)에서 exception이 발생하여 번거로우신가요?
소프트웨어가 실행되는 동안 exception을 추적하려면 많은 시간과 비용이 필요합니다.
하지만 T1.accessPredictor는 소프트웨어가 타겟에 플래시 다운로드 되기 전에
잘못된 메모리 접근 여부를 미리 확인할 수 있기 때문에 "오프라인 MPU"처럼 활용하실 수 있습니다.

AVOID MPU EXCEPTIONS

소스코드 분석? No! 바이너리 분석? Yes!

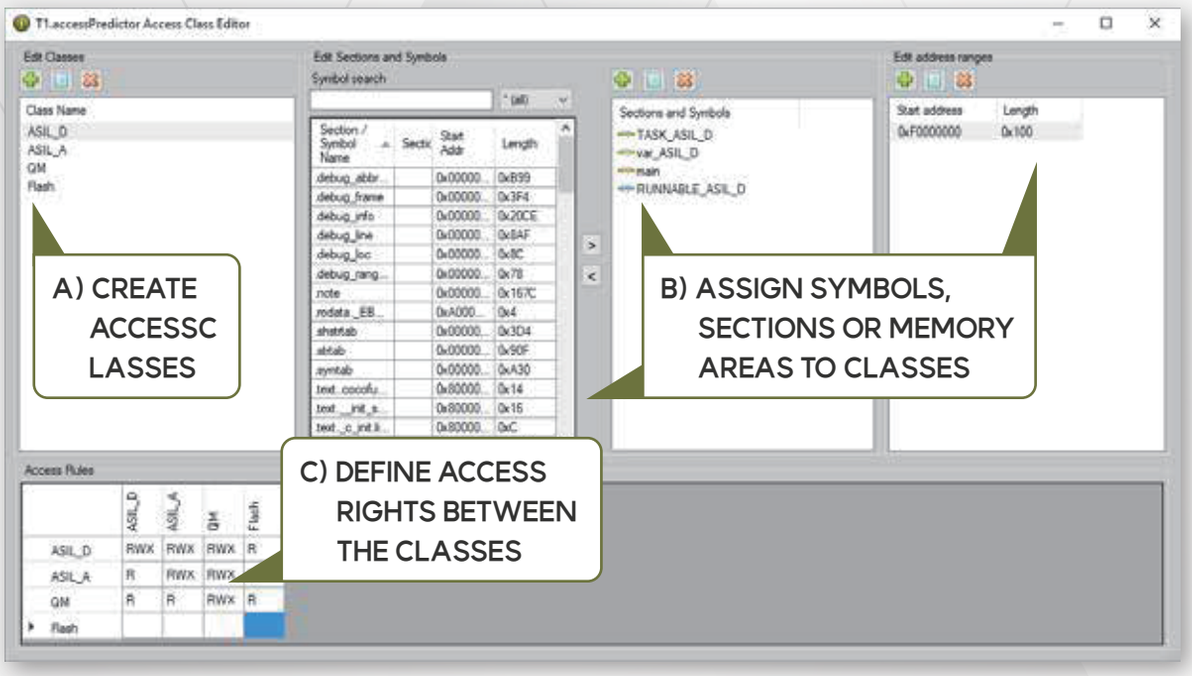
- 소스코드 분석은 컴파일러와 링커를 제외한 소프트웨어 원본 코드만 분석하고, 어셈블리 코드 분석도 생략하기 때문에 모든 문제점을 찾아내기 어렵습니다.
- ECU는 여러 개발자의 소프트웨어 컴포넌트를 통합하여 개발되기 때문에 어느 누구도 소스코드의 100%를 볼 수 없으며, 이는 소스코드 분석이 완벽하지 않다는 것을 의미합니다.
- 때문에 T1.accessPredictor를 이용하면 어셈블리 코드 분석을 통해 정확하게 잘못된 메모리 접근을 확인할 수 있습니다.





T1.accessPredictor는 사용 방법도 아주 간단합니다.

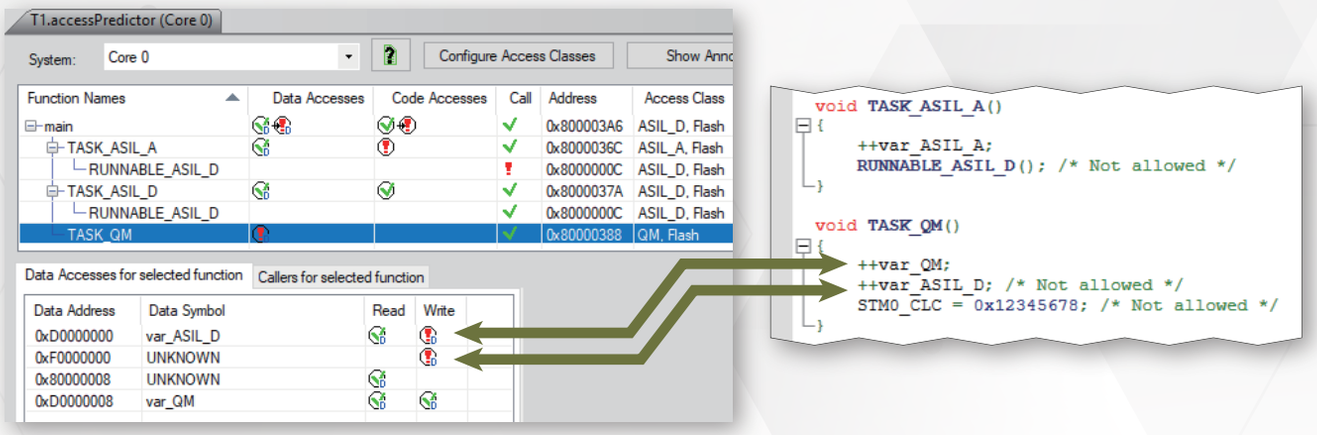
Step 1: 직관적인 GUI에서 다양한 액세스 클래스를 지정하고, 심볼/섹션/메모리 영역을 추가합니다.
다른 클래스에 접근할 수 있는 방식(Read, Write, eXecute)을 정의합니다.
"실행(eXecute)"은 함수 호출과 같은 코드접근을 나타냅니다.



Step 2: 바이너리인 ELF 파일을 읽습니다. T1.accessPredictor는 바이너리를 disassemble하고,
추상화(abstract interpretation)를 기반으로 정적 분석을 수행합니다.
그 후 T1.accessPredictor는 a) 어떤 함수가 다른 함수를 호출하고,
b) 다른 함수가 어떤 함수를 호출하는지 나타내는 "양방향" Call-tree를 제공합니다.

Step 3: 필요한 경우, Call-tree를 완성하기 위해 주석(수동, 생성, 측정 기반)을 추가합니다.

Step 4: 결과 분석 단계
Call-tree는 유효하지 않은 데이터 및 코드 접근에 대해 빨간색 느낌표로 위반을 나타냅니다.



Step 5(선택적) : 후속 소프트웨어 릴리즈에 대한 회귀 테스트를 진행하여 결과를 출력합니다.

경기도 용인시 기흥로 58-1
기흥ICT밸리 A동 1205, 1207호

itivai.com

+82-31-5174-9890
support@itivai.com