소프트웨어 검증의 어려움과 미래

권승완

2022.04.14

요 약

소프트웨어 검증은 소프트웨어의 어떤 속성을 수학적으로 증명하는 기술이다. 검증된 소프트웨어에는 반례가 없으므로, 소프트웨어의 안전성을 보장하는 가장 강력한 방법이라고 할 수 있다. 그러나 아직까지는 검증 가능한 소프트웨어가 매우 제한적이며, 검증하려는 속성이 소프트웨어의 안전함을 보이는 충분 조건인지 알지 못한다는 한계가 있다. 고가치 소프트웨어는 소유주가 공개 자체를 꺼린다는 문제도 있다. 사회가 디지털화 되어가는 것은 필연적인 변화일 것인데, 그 과정에서 검증의 수요와 필요성은 계속 증가할 것이다. 검증의 난제를 해결하고, 더욱 많은 소프트웨어를 검증 가능하게 하는 것이 안전한 미래 사회를 만드는데 큰 역할을 하리라 생각한다.

소프트웨어 검증이란 소프트웨어가 어떤 속성을 만족하거나 만족하지 않음을 수학적으로 증명하는 것이다. 증명이 완료된 소프트웨어에는 반례가 존재하지 않는다. 예컨데 소프트웨어를 실행하는 동안 널-참조 버그가 발생하지 않음이 증명되었다면, 그러한 버그를 일으킬 수 있는 입력은 존재하지 않는다. 이런 이유에서 소프트웨어 검증은 소프트웨어의 안전성을 보장하는 가장 강력한 방법이라고 할 수 있다.

그러나 검증은 몇 가지 한계를 가지고 있다. 먼저, 현재의 검증 기술로는 규모가 큰 소프트웨어를 검증하는 것이 불가능하다고 알려져있다. 소프트웨어란 프로그래밍 언어로 구현된 하나의 거대한 논리 함수이다. 따라서 소프트웨어를 검증한다는 것은 어떠한 전제 하에 소프트웨어라는 함수를 실행했을 때, 도출된 결론이 어떤 조건을 만족하는지 확인하는 것이다. 이때 만약, 소프트웨어 구현에 사용한 언어의 의미 구조가 복잡하여 소프트웨어를 정확하게 표현하기 어렵거나, 검증에 사용하는 논리 시스템의 한계로 인해 결론을 도출할 수 없다면 검증이 이뤄질 수 없다. 특히 괴델의 불완전성 정리에 따르면, 어떠한 논리 시스템에도 증명할 수 없는 참인 명제가 존재하므로 이러한 문제는 더욱 해결되기 어렵다.

다음으로, 검증해야하는 속성 자체에 대해 무지할 수 있다. 보안 분야에서 공격과 방어의 기술은 지속적으로 상호 진화해왔다. 스택 오버플로우가 등장하자 카나리가 도입되었고, 스택으로 실행 흐름을 옮기는 공격 기법이 개발되자 스택 및 기타 메모리에서 실행 권한을 제거하는 보호 기법이 도입되었다. 이후에는 실행 흐름 조작을 하나의 공격 기법의 대분류로 보고, 이를 막기 위한 다양한 모니터링 기술 및 운영체제 보호 기법이 개발되었다. 대표적으로 인텔의 제어흐름-무결성, 섀도우 스택 등을 비롯한 제어흐름 강화(Control-Flow Enforcement) 기술이 있으며, 주소 공간을 무작위로 할당하는 ASLR, PIE, KASLR 등이 있다. 이외에도 무수히 많은 기술들이 연구되어 왔다. 이러한 공방과정이 시사하는 것 중 하나는, 공격이 발생하기 전에는 시스템에 어떠한 공격 표면이 존재하는지 자체를 인지하기 어렵다는 것이다. 스택 오버플로우 기법이 등장한 이후에 스택을 보호해야함을 알았고, ptmalloc을 비롯한 메모리 할당자가 취약하다는 것이 알려진 이후 힙도 공격 대상이 될 수 있음을 알게 되었다. CPU에서 발생한 스펙터나 멜트오버, 자바스크립트 엔진에서 발생하는 타입 혼동(type confusion) 등도 마찬가지이다. 당연하지만, 우리는 어떤 공격 또는 공격의 유형을 알기 전에는 그것으로부터의 안전함을

검증할 수 없다. 그러므로 소프트웨어에 이미 알고 있는 보안 약점들이 존재하지 않음을 검증했다고 해서, 그 소프트웨어가 안전함을 증명한 것이 아니다.

마지막으로, 검증 대상의 획득이 어려울 수 있다. 이는 사실 검증 뿐만 아니라 프로그램을 분석하는 분야 전반에 해당하는 문제이다. 필연적으로, 검증이 필요할 정도로 가치가 높은 정보는 소유자가 공개 자체를 꺼린다. 특히 국방 소프트웨어, 행정 소프트웨어 등과 같이 법적인 규제를 받는 정보 자산은 외부로의 반출 자체가 불가능에 가깝다. 동형 암호가 이러한 정보 공유의 어려움 해소해줄 것으로 기대받고 있으나, 실용성을 확보할 때까지 얼마의 시간이 소요될 지 알 수 없다.

그러나 이러한 한계에도 불구하고, 소프트웨어 검증의 필요성과 수요는 계속 증가할 것이다. IoT를 얘기하던 것이 불과 10년 전인데, 지금은 메타버스, 디파이 등을 거론하며 디지털 대전환을 예고하고 있다. 이러한 키워드들이 모두 실현될 지는 모르겠으나, 우리 사회가 정보화의 방향으로 계속 나아갈 것은 부정하기 어려워 보인다. 따라서 앞으로는 지금보다 더 다양하고 많은 검증 수요가 생겨날 것이다. 이를 가늠할 수 있는 예로 디파이가 있다. 전 세계에 은행사는 많아야 수 백개이고, 사용하는 소프트웨어는 그보다 훨씬 적을 것이다. 그런데 디파이 프로토콜은 그 수를 헤아리기 어렵다. 또한 중앙 집중화된 시스템에서는 그를 운영하는 정보 주체를 비교적 쉽게 신뢰할 수 있었으나, 분산화된 시스템에서는 그를 신뢰하기 어렵다. 심지어 주체가 익명인 경우도 많다. '코드가 법이다'라는 말이 농담처럼 돌기도 하는 이유이다. 이러한 변화가 비단 금융에서만 나타나는 것이 아닐 것이다. 앞으로 디지털 전환이 이뤄지는 많은 분야에서 마찬가지 현상이 발생할 것이라 생각한다. 그래서 미래에는 검증이 단순히 소프트웨어를 증명하는 수단이 아니라, 소프트웨어와 시스템, 그리고 디지털 생태계를 신뢰할 수 있게하는 필수 요소가 되리라 본다. 검증된 디파이 시스템과 검증되지 않은 디파이 시스템이 있다면, 누구나 전자를 사용하지 않을까.