소프트웨어 보안기술 한차원 높이는 방법

이광근
서울대 컴퓨터공학부 교수
소프트웨어무결점연구센터 소장

디지탈타임스/[포럼]/2013.9.10

소프트웨어 보안은 일반 데이타 보안과 다르다. 비유하자면, 데이타 보안은 사랑하는 "애인의 손가방"을 지키는 일이고, 소프트웨어 보안은 그 "애인 자체"를 지키는 일이다. 손가방(데이터)은 완전히 수동적인 대상이지만, 반면에 애인(소프트웨어)은 움직이고 판단하고 결정하는 자율이 있는 다이나믹한 대상이다.

데이터 보안은 완전히 수동적인 데이터를 지키려는 기술이다. 데이터를 설사 누가 훔쳤더라도 그 내용을 절대 알아볼 수 없게 하는 기술이다. 시스템 기술을 통해서 데이터 접근 권한을 제한하고, 암호화 기술을 통해 데이터 내용을 완벽히 감추는 기술이 이에 해당한다.

반면, 소프트웨어 보안은 다이나믹하고 자율적인 소프트웨어를 지키려는 기술이다. 소프트웨어는 실행하면서 살아움직이도록 만들어진다. 외부와 소통하고, 끊임없이 논리적인 판단을 하며, 무한히 많은 외부 자극들에 모두 제대로 반응하며 중단없이 진행해가야 한다. 소프트웨어 보안 헛점이란 소프트웨어가 위와같은 실행중에 의도적으로 나쁜 일을 하거나, 외부의 나쁜 꾀임에 빠져들어 의도치않게 나쁜 일을 하게되는 것을 말한다. 소프트웨어 보안 기술은 이러한 헛점들을 탐지하고 방지해 주는 기술이어야 한다.

당연히, 소프트웨어 보안 헛점은 미연에 방지해야 할게고, 다행히도 미연에 방지할 방법이 있다. 소프트웨어는 사람과 달리, 자신의 모든 자율 실행과정이 소프트웨어의 소스에 고스란히 표현되어 있다. 따라서, 소프트웨

어 소스를 잘 분석할 수 만 있으면, 그 소프트웨어의 보안 헛점을 미리 모두 찾아낼 수 있다.

그러나 이런 헛점을 찾는 것은 쉽지않다. 소프트웨어 소스의 크기가 웬만한 대하소설보다 크고, 그 논리의 흐름의 복잡도는 포유류 뇌 속의 뉴런들의 연결관계만큼 복잡하다. 이때문에 소프트웨어를 만드는 사람은 늘 실수할 수 밖에 없고, 이러한 실수들이 소프트웨어 보안 헛점으로 어느 구석엔가 남아있게 된다. 이렇게 숨어있는 보안 헛점을 찾는 것은 해운대 모래사장에서 잃어버린 샤프심을 찾는 것과 비슷하다.

더욱 어려운 점은, 보안 헛점을 찾는 소스 분석은 단순히 소스 텍스트의 겉모양만 훑어서는 안된다는 것이다. 컴퓨터가 그 소스를 실행하면서 하는 일들(의미)를 깊이 분석할 수 있어야 한다. 단순히 소스 텍스트의 겉모양만 가지고 찾을 수 있는 헛점은 소프트웨어 보안의 핵심 헛점이 아닌 경우가 대부분이다. 비유하자면, 생긴것 만 보고는 사람의 진면목이 드러나지 않는 것과 같다. 실제 그 사람 두뇌속 뉴런의 연결관계들이 실행되는 모든 시나리오들(소프트웨어 소스의 실행의미)을 예측해 낼 수 있어야 한다.

이러한 깊이있는 분석 기술을 의미기반 분석기술(semantic-based static analysis)이라고 한다. 단순한 구문기반(syntactic-based, 겉모양만 보는) 분석기술의 명백한 한계를 뛰어넘는 기술이다. 학계에서 산업계로 기술이전까지 종종 되어온 선도적인 기술이다. 그리고, 이런 의미분석을 제대로 수행하는 제품을 만드는데는 프로그래밍언어 분야의 첨단 이론을 이해하는 것이 뒷 받침되야 하기때문에 진입 장벽이 높다. 이론의 깊이가 없이 1-2년 흉내내서는 제대로 상용화하기에는 한계가 있는 분야이다.

근래에 소프트웨어보안에 대한 관심이 매우 높아지고 있고, 최근에 정부는 시큐어코딩의 의무화 등을 통해 정부가 사용할 소프트웨어의 보안 요구사항들을 엄격히 규정하고 있다. 시큐어코딩의 의무화에는 소프트웨어 소스의 겉모양뿐 아니라 실행되는 속 내용에 대해서 보안 헛점이 없도록 하는 광범위한 요구사항들이 모두 포함된다.

이러한 보안 요구사항을 만족시키는 지를 검수하는 데 제대로된 의미기반 분석기술이 사용되도록 유도하는 것이 선진국 정부다운 소프트웨어 보안 정책이 될 것이다.