정적 분석과 인공지능(AI)의 발전: 개발의 패러다임을 바꾸다

이정재

최근 IT 업계에서는 코드 품질 향상과 개발 효율성 극대화를 위한 정적 분석(Static Analysis) 기술의 중요성이 점점 커지고 있다. 단순한 문법 오류 탐지를 넘어 보안 취약점까지 사전에 발견하기 위해, 많은 기업이 정적 분석을 필수 개발 프로세스로 도입하는 추세다. 예를 들어, Google은 코드 리뷰 과정에서 정적 분석을 자동으로 수행하는 'Tricorder' 시스템을 적용해 개발자들의 신뢰를 얻었으며, Facebook 또한 'Infer'와 'Zoncolan' 같은 도구를 활용해 보안 취약점 탐지 기능을 지속적으로 강화하고 있다.

완벽한 코드를 처음부터 작성하는 것은 현실적으로 어렵다. 복잡한 로직을 구현하는 과정에서 실수는 불가피하며, 때때로 이러한 실수가 치명적인 프로그램 오류로 이어지기도 한다. 개인적으로도 나 자신을 스스로 돌아봐도 사소한 오류 하나가 예상보다 많은 시간을 소요하게 만든 경험이 많다. 심지어 그 실수를 했던 부분이 아주 간단한 문제였던 경우인 적이 태반이다. 예를 들면, 예상치 못한 메모리 누수($memory-leak$)나 무한 루프($infinite-loop$)로 인해 코드 구조를 재설계해야 했던 적이 있었다. 물론 개발 과정에서 시행착오는 필연적이지만, 정적 분석 도구를 효과적으로 활용하면 이러한 실수를 사전에 방지할 수 있으며, 결과적으로 개발자의 생산성을 크게 향상시킬 수 있다고 생각한다.

정적 분석은 프로그램을 실행하지 않고도 코드 자체를 분석하여 잠재적인 문제점을 찾아내는 방식으로, 실행 중 오류를 감지하는 동적 분석($Dynamic-Analysis$)과 차별화된다. 일반적인 컴파일러가 단순한 문법적 오류를 잡는 것과 달리, 정적 분석 도구는 코드 패턴을 기반으로 잠재적인 보안 취약점과 논리적 오류까지 탐지할 수 있다. 이러한 개념은 개발자들이 경험을 공유하며 더 나은 코딩 습관을 형성해 나가는 과정과 유사하다. 예를 들어, 숙련된 개발자들은 이전 프로젝트에서 발생했던 버그 패턴을 기억하고, 새로운 코드에서도 같은 실수를 방지하려는 노력을 기울인다. 정적 분석 도구도 마찬가지로 다양한 코드 패턴을 학습하고, 잠재적인 문제를 사전에 탐지하도록 설계된다. 결국, 이는 인간의 경험을 기반으로 한 자동화된 코드 리뷰 시스템이라 할 수 있다. 다만 그 자동화됨에 있어서 정교한 수식과 수학적 정의로 채워져있으며 정적 분석 장치들은 오류에 대해서 충분히 설명 가능하다고 생각된다.

그러나 정적 분석 도구가 실용적으로 활용되려면 정탐($True-Positive$)과 오탐($False-Positive$) 비율이 적절하게 균형을 이루어야 한다. 탐지 범위를 지나치게 넓히면 오탐이 많아지게 된다면 자연스레 그 정적 분석 장치의 신뢰는 잃게 될 것이다. 당연하다고 생각한다. 양치기 소년과 같은 사례처럼 개발자들이 경고를 무시하게 되고, 반대로 지나치게 좁히면 중요한 오류를 놓칠 가능성이 커진다. 실제로 에세이를 쓰기 위해 읽었었던 자료에서 Google이 초기 도입했던 FindBugs는 높은 오탐 비율 때문에 개발자들에게 외면받은 사례가 있다. 따라서 정적 분석 도구가 실질적으로 활용되려면, 분석 결과가 개발자의 자연스러운 워크플로우에 녹아들어야 하며, 필요한 순간마다 즉각적이고 정확한 수준의 피드백을 제공할 수 있어야 한다.

추가적으로 정적 분석의 글을 읽으면서 인공지능(AI)를 이용한 정적분석에 대한 여러가지 생각이 들었다. 특히 거대언어모델(LLM) 기반의 자율 에이전트($Autonomous-Agent$) 기술이 급격히 발전하면서, 정적 분석에도 새로운 가능성이 열리고 있다고 생각된다. 인공지능을 활용한 코드 분석 시스템이 개발자의 실수를 사전에 감지하고, 자동으로 수정 제안을 제공하는 방향으로 발전하고 있다. 인공지능 기반 코드 분석 시스템은 개발자의 코드 작성 패턴을 학습하고, 과거의 실수와 비교하여 오류 발생 가능성을 예측하는 기능을 수행할 수 있다. 이는 단순한 정적 분석을 넘어 맥락을 고려한 코드 리뷰와 리팩토링($refactoring$) 추천까지 가능하게 만든다.

현재 GitHub Copilot, Amazon CodeWhisperer와 같은 인공지능 도구들은 이미 코드 자동 완성 및 오류 수정 기능을 지원하고 있으며, 향후 더 발전된 인공지능 모델이 정적 분석에 접목될 가능성이 크다. 인공지능이 코드 작성 과정에서 자동으로 보안 취약점을 감지하고, 리팩토링을 제안하는 기능을 갖춘다면 보다 안전하고 최적화된 코드 작성이 가능할 것이다. 그러나 현재의 인공지능 모델이 모든 프로그래밍 작업을 완벽하게 수행하기에는 아직 한계가 있다. 인공지능이 분석할 수 있는 코드의 크기와 복잡성에는 제약이 있으며, 모델이 학습한 데이터에 따라 편향된

결과가 나올 수도 있다. 따라서 인공지능 기반 정적 분석 시스템이 실용적으로 활용되려면, 지속적인 개선과 학습이 필수적이다. 다만 이런 인공지능 도구가 놀랍다고 생각되는 부분은 여러개의 하위 파일들로 구성되어 있는 전체 프로그램에서의 개발자의 의도를 파악해서 코드를 실행하지 않고 정적으로 분석이 가능하다는 것이다. 이는 기존의 정적 분석과는 다른 확장의 개념인 것 같다. 정확한 문제를 찾고 설명하고자 한다고 볼 수 있는 정적 분석과는 다르게 사용자의 관점에서 실시간으로 실행하지 않고 내 코드를 정적으로 훑어서 검사를 해주는 것이다. 의도를 고려한 정적 분석인 셈이다. 해당 기술이 정적 분석이라고 할 수 있는지는 아직까지는 잘 모르겠다. 하지만 이런 반복적인 코드 리뷰나 버그 수정 같은 작업을 자동화하는 것도 매우 흥미로운 주제다. 만약 자율 에이전트가 단순 반복적인 코드 수정 작업을 대신 수행한다면, 개발자들은 보다 창의적인 문제 해결에 집중할 수 있을 것이다. 이는 정적 분석의 효과를 극대화할 뿐만 아니라, 코드 품질 향상과 개발 속도 개선에도 기여할 수 있을 것이라고 생각한다.

또한 나는 최근 자율 에이전트의 행동을 탐지하는 과정에 대해 연구하고 있는데, 흥미롭게도 정적 분석에서 문제가 되었었던 '오탐과 정탐 비율'의 문제는 자율 에이전트 탐지 과정에서도 동일하게 적용될 가능성이 크다고 생각된다. 자율 에이전트의 동작 과정에서는 적정한 검증단계가 필요하다. 에이전트가 사용자의 의도와 다르게 이상한 행동으로 물건을 결제하거나 중요한 문서를 삭제해버리면 안되기 때문이다. 하지만 이런 검증은 단순한 패턴 인식만으로는 충분하지 않다. 왜냐하면 실제 환경에서 다양한 변수를 고려해야 하기 때문이다. 하나의 사용자 명령을 수행하는 방법은 굉장히 여러가지가 있다. 메인 화면을 거쳐서 메세지를 보내는 경우도, 알림을 눌러 메세지를 보낼 수도 있다. 그렇기 때문에 에이전트와 정상적인 행동과 악의적인 행동을 구별하는 과정에서는 단순한 코드 패턴이 아니라 실행 동작 과정 전체에서의 맥락(*Context*)을 고려한 탐지 알고리즘 또는 검증 방식이 필요하다. 하지만 그 고려 검증 방식이 너무 강하다면 에이전트는 자유로운 사용자의 명령 수행이 제한이 될 것이다. 적절한 탐지 비율을 설정하여 대부분의 경우를 효과적으로 다룰 수 있도록 하는 것이 핵심이 될 것이라고 생각한다.

특히 모바일 환경에서 거대언어모델(*LLM*) 기반의 에이전트는 더욱 복잡한 도전 과제에 직면한다. 모바일 애플리케이션의 UI 사용의 자동화를 위해 인공지능 에이전트는 유저의 명령을 해석하고, 이를 수행하기 위해 여러 단계의 동작($Click-Action, Scroll-Action, Input-Action$)을 계획해야 한다. 이때 분석이 개입할 수 있는 시점은 크게 두 가지로 나뉜다. 첫째, 초기 명령어를 받는 순간 정적 분석을 통해 잠재적인 보안 문제나 실행 오류를 사전 감지할 수 있다. 예를 들어, 인공지능이 특정 UI 버튼을 클릭하려 할 때, 해당 버튼이 비활성화 상태이거나 보안상 민감한 동작을 유발하는지 사전 확인하는 방식이다. 둘째, 실행 중간에 동작의 연속성을 점검하여 예상치 못한 오류나 보안 위험을 실시간으로 분석할 수도 있다. 예를 들어, 앱 내에서 반복적인 클릭 패턴이 의도된 사용자 동작인지, 악성 행위인지 분석하는 과정에서 분석이 필요하다고 볼 수 있다. 전자는 정적 분석에 해당하고 후자는 동적 분석에 해당한다. 특히 정적 분석 단계에서 모든 검증 부분이 확인된다면 인공지능 에이전트의 동작은 안전하게 보장될 수 있을 것이라고 생각한다.

나는 작년에 정적 분석을 처음 접했을 때 이에 대한 글을 작성한 적이 있다. 당시에는 정적분석이 단순한 오류 탐지 도구 정도로 생각했지만, 다시 그 글을 읽어보니 느낌이 사뭇 다르다. 그때는 정적 분석이 무엇인지 이해하는 데 집중했다면, 지금은 인공지능와 융합된 정적 분석이 개발의 패러다임을 어떻게 변화시킬 것인지에 대해 고민하고 있다. 기술이 빠르게 변화하고 있는 만큼, 나의 시각도 계속해서 변화하고 있음을 실감하는 순간이다.