효율성이 주는 착각: AI 코딩의 보안 위협과 한계

KAIST 전산학부 한승우

본 글은 AI 기반 코딩 도구의 확산을 중심으로, 개발 속도의 향상과 함께 따라오는 보안 취약성과 오류 유발 가능성을 비판적으로 살펴본다. 또한 AI가 생성한 코드가 문법적으로는 완벽해 보일 수 있으나 실제로는 여러 공격에 취약하거나 치명적인 오류를 포함할 수 있음을 지적한다. 이러한 도구에 대한 무비판적인 신뢰는 소프트웨어의 안정성과 신뢰성을 위협하며, 인간 개발자의 철저한 검토와 보안 점검 없이는 위험이 가중될 수 있다. 정적 분석 도구 적용 등의 체계적 검증 절차가 필수적이며, AI가 제공하는 결과에 대해 지속적인 경계와 비판적 시각이 요구된다. AI가 만들어낸 편리함 이면에 숨어 있는 위험을 직시하고, 인간의 적극적인 개입을 통해 이를 통제해야 한다.

AI는 이제 코드를 짠다. 자동 완성에서 시작했던 것들은 이제 GitHub Copilot이나 Amazon CodeWhisper와 같은 AI 코딩 도구로 변모하였다. 이 도구들은 함수를 작성하고, 코드베이스 전체를 정돈(refactor)하며, 보안 위험이 있는 코드를 고치기도 한다. 하지만 이 변화에는 위험이 따랐다: Wired 및 CACM에 발표된 연구에 따르면 AI 생성된 코드에는 사소한 버그와 안전성 결함이 높은 비중으로 포함되었다 [1, 4]. AI는 개발 속도를 제공하지만, 그 속에는 위험 요소 또한 실재한다.

처음으로 Copilot을 사용했을 때 마치 불법적인 일을 하는 듯한 죄악감이 느껴졌다. 간단한 웹 서비스를 만들 일이 있었는데, 간단한 영어 몇 문장의 설명만으로 Copilot에 의해 웹페이지 로그인 로직이 순식간에 실제화되었다. 그 코드는 잘 동작했고, 올발라 보였다. 하지만 며칠 후 코드를 다시 보면서 '실제 서비스에 Copilot을 쓰지 않아 다행이다'라고 생각할 만한 일이 일어났는데, Copilot이 제공한 코드가 SQL 삽입 공격에 취약했기 때문이다. 그 순간 마음속에서 신기술은 위협 인자가 되었다.

AI는 개발을 가속화하는 동시에 코드의 불안정성을 퍼뜨린다. AI는 결국 학습 데이터에 의존한다. IEEE의 한 기사는 Codex와 같은 AI 모델은 수십억 줄의 오픈 소스 코드로 학습되었고 그 코드들의 많은 부분에 흠결이 있다고 말한다 [5]. 이런 방식으로 만들어진 AI 코딩 도구들은 결국 '사람처럼' 오류가 있는 코드를 작성할 수밖에 없다는 것이다. 뉴욕대의 연구를 인용한 Wired의 기사에 따르면, 보안이 중요한 프롬프트의 40%에서 Copilot은 안전하지 못한 코드를 출력하였다 [1]. 40%의 수치는 이런 문제가 일부 특수한 경우에서뿐만 아니라 일반적으로 나타난다는 것을 보여준다.

AI의 사용은 '안주安住의 대가'를 요구할 수 있다. AI가 작성한 코드의 안전성 문제는 항상 눈에 보이지는 않는다. 한 Nature의 사설에 따르면, 전문가들은 Excel과 같은 소프트웨어의 기능이 생산성 향상에 도움이 될 수 있지만, 정확성에 대한 '잘못된 착각'을 유도할 수 있다고 경고한다 [3]. 실제 현장에서 '잘못된 착각'이란 잘못 적용된 암호화, 확인되지 않은 입력, 약간 이상한 로직을 의미할 것이다. 한 ACM의 기사는 AI가 생성한 코드가 입력을 제대로 검증하지 않아 다양한 종류의 삽입(injection) 공격을 허용하는 경우가 흔하다고 말한다 [4]. AI가 코드 작성의 비용을 절감할 수 있어도 결국 AI가 초래하는 보안 위협에 대한 '안주 비용'을 증폭할 것이다.

Copilot은 반박하지도, 머뭇거리지도, 의심을 표현하지도 않으며, 다만 문법적으로 완벽해 보이는 코드로 응답한다. 이는 잘못된 신뢰를 낳는다. 개발자들은 일관성 있고 그럴 듯해 보이는 출력물에 신뢰를 줄 것이다. AI가 위험해지는 지점은 틀렸을 때가 아니라, '조용히' 틀렸을 때이다. 견고한 검증 체계 없이는 모든 것이 사상누각일 뿐이다.

우리는 코딩 AI에 대한 신뢰의 분수령에 서 있다. 인간의 개입 없이 코드 작성, 검토 및 배포를 완료하는 완전 자율 시스템을 적용할 수도, 철저한 인간 감시, 능동적 보안 감사, AI 코딩의 구조적 경계가 필요한 시스템을 선택할 수도 있다. 전자는 효율적이지만 불투명하고 빠르지만 취약하다. 후자는 비효율적이지만 투명하고 느리지만 안전하다. 기존 AI 학습 방법과 기존 소프트웨어 오류 분석기를 결합하여 견고한 AI 코딩 시스템을 만들려는 시도도 계속되고 있다

[2]. 하지만 이러한 AI 코드의 안전성을 보장하기 위한 노력보다 AI로 자동화되는 속도가 더 빠른 것이 현실이다.

"우리는 그저 지켜볼 것인가?" 이는 모든 AI 도구가 묻는 암묵적 질문이다. 수동적인 사용은 곧 동의이며, 그 대안은 '경계警戒'이다. 인간의 코드 리뷰, AI 코드에 대한 레드 팀 테스트, CodeQL과 같은 정적 분석 도구, 그리고 편리함의 유혹에 대한 저항이 AI가 주는 위협으로부터 우리를 보호할 것이다. AI는 프로그래머를 대체하지 않고, 대체하지 못할 것이다. 하지만 AI는 게으른 프로그래머가 더욱 큰 문제를 더욱 빠르게 일으키도록 유도할 것이다.

AI는 결코 코딩을 멈추지 않을 것이다. 우리 또한 코딩을 멈춰선 안 된다. 빠른 속도는 철저한 검토와 균형을 이루어야 한다. 신뢰는 주어지는 것이 아니라 증명되어야 한다. 이제 보안은 좋은 코드를 짜는 것뿐만 아니라, 기계가 제공하는 코드를 잘 감시하는 것 또한 요구한다. AI와 함께 코딩할 미래는 이미 주어져 있고, 유일한 질문은 누가 결과물을 검증할 것인지이다.

References

[1] Douglas Heaven. 2021. AI Can Write Code Like Humans—Bugs and All. *WIRED* (2021). https://www.wired.com/story/ai-write-code-like-humans-bugs/

[2] Kihong Heo. 2022. Trustworthy AI. https://prosys.kaist.ac.kr/trustworthy/

[3] Dyani Lewis. 2021. Autocorrect Errors in Excel. *Nature* (2021). https://www.nature.com/articles/d41586-021-02211-4

[4] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. 2025. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. *Commun. ACM* (2025). https://cacm.acm.org/research-highlights/asleep-at-the-keyboard-assessing-the-security-of-github-copilots-code-contributions

[5] Craig S. Smith. 2022. Coding Made AI—Now, How Will AI Unmake Coding? *IEEE Spectrum* (2022). https://spectrum.ieee.org/ai-code-generation-language-models