끊기지 않는 버그, 그리고 끊기지 않는 맞섬

문성민

프로그래밍에서 버그는 떼놓을 수 없는 문제입니다. 컴퓨터가 발명되고 프로그래밍의 개념이 생긴 순간부터 프로그램 버그라는 문제는 언제나 함께 존재해왔고, 이를 해결하는 것이 전산학에 있어서 궁극적인 목표 중 하나로도 볼 수 있다고 생각합니다. 그러나 대부분의 버그는 사람의 실수로부터, 바꿀 수 없는 요소로부터 그 원인이 유래합니다. 그에 반해 컴퓨터가 사용자의 프로그램 구조를 스스로 파악하는 능력이 현재로서는 아직 매우 미흡하기에, 프로그램 버그라는 요소는 해결하기 너무나도 어려운 문제로 자리 잡고 있습니다. 하지만 그 궁극적인 해결까지 닿지 못하더라도, 이를 완화하기 위한 다양한 방법론적인 시도가 이루어지고 있으며 현실적인 측면에서도 시장 원리에 따라 회사들과 개발자들은 치명적인 버그들을 결국 없애기 위해 분주히 움직이고 있습니다. 버그의 완전한 해결이라는 궁극적인 목표에는 도달할 수 없더라도, 그에 방향을 두고 있는 수많은 도전에 대해 값진 가치가 있다는 것 또한 우리가 생각해볼 수 있는 사실일 것입니다.

2024년 4월, 세계적으로 수많은 사람이 이용하던 메신저 앱인 텔레그램에서 치명적인 보안 문제가 발견되었던 사건이 있었습니다. 외부인이 악성 코드를 제공하면 이를 텔레그램 앱의 취약점을 이용해 원격에서 실행시킬 수 있게 되는 형태의 문제점이 발견되었던 것인데요. 확인 결과 이 치명적인 문제는 개발자의 프로그래밍 도중 발생한 사소한 오타로부터 비롯된 것이었습니다.

같은 해 7월에는 Window 10의 보안 시스템의 업데이트 후 블루 스크린이 발생하여 분야와 나라를 불문하고 세계적으로 막대한 피해가 발생한 사태가 있었습니다. 이는 Window 10 시스템의 보안 프로그램을 개발하는 회사에서 하자가 있는 파일을 배포하여 생긴 문제로 밝혀졌습니다.

컴퓨터라는 놀랍도록 유용한 도구가 발명된 이후, 세상에는 수많은 프로그램이 개발되어왔습니다. 앞선 사건들을 포함해 그 프로그램들에 내재한 수많은 문제, 즉 버그 또한 발생했죠. 이렇게 프로그램과 관련된 치명적인 문제들을 두 부류로 나누면, 하나는 외부인의 공격 때문에 어떤 정보에 대해 허락되지 않은 접근이 일어나는 것입니다. 다른 하나는 프로그램 자체로부터 그 기능에 문제가 발생해 제대로 작동하지 못하거나, 혹은 전자의 경우에 대해 대비하는 "방어"가 기능하지 못해 잠재적인 위험을 품게 되는 것입니다. 저는 이 둘 중 후자에 집중해보고자 합니다.

컴퓨터가 발명된 이후로 현재까지 컴퓨터 그 자체의 기능에는 굉장한 발전이 있었고, 이를 따라 프로그래밍에 대한 방법론 또한 시간이 지남에 따라 크게 변화해왔습니다. 더 간편해지고, 효율적으로 변했습니다. 다만 극도로 발전하고 변화한 컴퓨터와 달리, 그 컴퓨터를 사용하는 우리 사람들에게는 이에 비례하는 변화가 있었다고 보기에는 어려울 것입니다. 사람들은 컴퓨터를 이용해 프로그래밍하며 수많은 실수를 해왔습니다. 프로그램을 작성하는 도중 크고 작은 오타를 내고, 종종 프로그램의 구현 도중 논리적으로 오류가 있는 아이디어를 사용하여 문제가 발생하기도 합니다. 이는 앞에서 언급한 두 가지 문제 중 후자의 직접적인 원인이 됩니다. 그리고 이는 어찌 보면 컴퓨터의 발전만으로는 근본적으로 해결될 수 없는 문제일 것입니다. 순수히 이를 사용하는 사람으로부터 문제가 발생하는 것이니까요.

그렇다면 앞으로 이런 측면은 미래에 어떻게 변화할 것이며, 이를 해결하거나 최소한 완화하기 위해 앞으로 어떤 시도가 있을 것인지 고민해보아야 합니다. 프로그램 버그에 대한 문제가 미래에 어떻게 변할지에 대해 생각한다면, 저는 앞으로 현재와 같은 상황이 지속할 것으로 생각합니다. 앞으로도 새로운 프로그램들이 생산되며 버그는 계속해서 생겨날 것이고, 이미 존재하던 버그들 위에 누적될 것입니다. 사람이 프로그래밍 도중에 크고 작은 실수를 하는 것은 바꿀 수 없는 데에 반해, 그런 실수를 컴퓨터가 직접 정밀하게 바로잡기에는 전체 프로그램의 구조를 해석하는 능력이 컴퓨터에는 매우 부족한 상황입니다. 그렇기에 사람과 컴퓨터 사이 상호 보완적인 과정을 통한 완화는 가능하더라도, 완전한 해결은 현재로서는 어려울 것입니다.

사람의 실수로부터 발생하는 버그들을 해결하는 데에 그 목표를 크게 두 가지로 분류하자면 하나는 기존에 발생했던 버그를 고치는 것, 다른 하나는 현재진행형으로 발생하는 버그를 줄이는 시도일 것입니다. 후자의 경우, 즉 새롭게 발생하는 버그를 줄이기 위한 방법론적인 시도는 현재 다양한 방식으로 이루어지고 있으며, 앞으로도 계속되어 지속적으로 문제의 완화를 이루어줄 것입니다. 기존의 프로그래밍 언어와 다른, 혹은 더 엄격한 규칙을 가지며 사용자의 실수를 줄이는

데에 목적을 두는 다양한 함수형 프로그래밍 언어가 새로이 등장하고 있음을 그 예로 들을 수 있을 것입니다. 이뿐만 아니라 자동화된 안전성 검증 등에 관한 연구도 현재 활발히 이루어지고 있습니다. 이는 사용자가 구성한 프로그램의 구조를 컴퓨터가 스스로 해석하는 능력을 발전시킴으로써 문제를 해결하기 위한 시도로 볼 수 있습니다. 이러한 개발이 지속하며, 컴퓨터 속 버그의 문제가 완전히 사라질 수는 없더라도 서서히 이를 완화하고 우리가 스스로 감당 가능한 수준에 이를 수 있다고 기대합니다.

반면에 기존에 발생했던 버그를 해결하는 문제에는 저는 현재 상황과 같이 시장 원리에 기대어야 한다고 생각합니다. 이미 누적된 버그의 수가 셀 수 없이 많고, 이를 완벽히 해결하는 것은 인력과 자원의 부족으로 불가능할 것이 분명합니다. 그렇기에 만약 버그를 해결하기 위한 자원이 있다면, 그 수많은 문제 중에서 각 버그의 중요성에 비례하여 작업해주는 효율적인 과정이 필요할 것입니다. 이때 우리가 참고해야 할 점은, 대체로 이러한 문제들의 중요성이 각각 그 문제를 다루는 개발자 및 회사의 이익과 직결된다는 사실입니다. 분야와 환경의 특성상 중요도가 높은 오류는 곧 잠재적인 위험이 큰 오류이기에 공급자의 이익에 효과적으로 반영될 것이고, 이러한 측면에서 저는 외부의 개입이 없는 환경만으로도 충분히 효과적인 문제 해결이 될 것으로 생각합니다.

프로그램 개발에 있어서 버그는 언제나 문제가 되어왔습니다. 이를 완벽히 해결하는 것은 적어도 가까운 미래 안에는 불가능할 것이나, 그 문제를 조금이라도 더 약화시키기 위한 시도는 오래전부터 있었습니다. 앞으로도 다양하고 새로운 형태로서 프로그램 버그에 맞서는 수많은 시도가 있을 것이고, 이는 유의미한 변화로서 프로그래밍의 방법론과 더 나아가 그 미래를 바꿀 것입니다. 비록 그것으로도 현실적으로 해결하기 어려운 수많은 문제가 남아있지만, 끊임없는 도전으로 이를 넘어서는 때가 오기를 기대합니다.