論文の内容の要旨

論文題目: Implementation of a Fail-Safe ANSI C Compiler
安全な ANSI C コンパイラの実装手法

氏名: 大岩 寛

C 言語で書かれたプログラムは、迷子ポインタやバッファ溢れなどによる厄介なバグの影響を受けがちであることはよく知られている。とりわけ、インターネット上のサーバプログラムにおけるそのようなバグは、悪意の攻撃者によってシステム全体を乗っ取るための攻撃の対象となりがちで、最近では社会的な問題にすらなっている。このような厄介なバグは元をたどれば、メモリ上の配列の境界を越えたアクセスにより、データ構造が破壊されることである。最近の言語、例えば Java、C#、Lisp、ML などの言語はこのような境界を越えたアクセスに対して保護機構を用意しているが、C 言語にはそのような機構はない。しかし、これは C 言語のデザイン上の欠陥とは言えない。なぜなら、C 言語は元々アセンブラ言語の置き換えとして、つまりは柔軟で直接的なメモリ操作を高級言語で記述するためにデザインされたものだからである。言い替えれば、このような保護機構の欠如は「わざと」導入されたものである。また、C 言語がデザインされた 30 年前には、当時の計算機能力に対して、このような保護機構を導入するのが現実的でなかったという点もある。過ちとされるべきはむしろ、そのような C 言語を現代の日常のプログラミング言語として、実際には直接的なメモリアクセスが必要とされない場合にも用いていることにある。けれども今日において、C 言語を直ちに放棄してしまうことは現実的ではない。C 言語で書かれた既存のプログラムは多く存在し、また C 言語やそのプログラミングスタイルに慣れ親しんだ「既存のプログラマ」も数多いからである。

このようなジレンマを解決するために、C 言語を安全に実装する多くの試みが提案され実際に実装されてきた。しかし、我々の知る限りそれらのすべては、危険な操作の全てを拒否し、**同時に**全ての ANSI C のプログラムを処理できるという目標を達成していない。C. Cowan による StackGuard に代表される実装のグループは、場当たり的な検査手法でプログラムに出現する特定の形の誤りを検出するだけのものであるし、他方 SafeC に代表されるグループは、C 言語の仕様の一部分のみを入力として受け付けるものである。G. Necula によって提案されている CCured が、我々の知る限りでは現時点でもっとも目標に近いものであるが、これも完璧であるとはいえない。

本論文は、この問題に対するもっとも強力な解を提案する。本論文で提案する Fail-Safe C

は、メモリ安全な ANSI C の完全な実装である。この実装は、全ての危険な操作を禁止しつつ、キャストや共用体を含む全ての ANSI C 標準に準拠し、かつ ANSI C の範囲を越えたプログラムに頻出するいわゆる「汚いトリック」の多くをも許容する。同時に、本実装は、コンパイル時と実行時双方で行なわれるさまざまな最適化によって、実行時検査の負荷の削減をはかっている。Fail-Safe C コンパイラを用いることで、プログラマは簡単に、自らの書いたプログラムに変更を加えることなしに、また移植作業をすることなしに、安全に実行することが可能となる。論文中では、実在する有名なプログラムに存在するセキュリティー上の脆弱性を用いて、実際に Fail-Safe C を適用して安全性を保証する実験を例示している。

この論文で述べられているいくつかの重要なアイディアは以下の通りである。

- メモリブロックの特殊な表現により、動的な境界検査と型検査を実現すること、
- オブジェクト指向の概念を用いてメモリブロックを表現し、全てのメモリブロックにアクセスメソッドを付加することにより、ポインタのキャストなどの静的型によらないアクセスの安全な実行をサポートすること、
- 「virtual offset」と名付けたメモリのアドレスづけの特殊な方法により、既存のプログラムの互換性の向上とキャスト操作の安全性を同時に実現していること、
- そして、ポインタがキャストされているかどうかを自らに記録するような、ポインタ(と整数)の賢い表現により、安全にキャストを実装すると同時に通常のポインタの高速な使用を実現したこと。

Fail-Safe C の環境下では、プログラム中の値がポインタとして参照に用いられるたびに、参照先ブロックのサイズと型との整合性を検査される(コンパイラが検査を省いても安全であることを確実に判定できた場合を除く)。ポインタが参照先ブロックのサイズを超過したメモリを参照している場合、実行時エラーが報告されプログラムは直ちに停止される。ポインタの型と参照先ブロックの型が整合しない場合は、アクセスハンドラメソッドが参照に用いられ、プログラムの実行の安全性を保証する。どちらでもない場合は、プログラムが直接メモリを参照することで、高速な実行を実現する。ポインタがキャストされたか否かの情報は、コンパイラによって正確に維持され、ポインタの型整合の判定を高速に行なえるようにしている。また、virtual offset の概念は、先に述べた一連の動作をプログラムから隠し、「舞台裏でこっそり行なわれるもの」にする。つまり、実行中のプログラムは、Fail-Safe C の監視下で実行されているということを認知することは、安全でないプログラムが突然終了させられることを除いてはできない。このことは、さまざまな「汚いトリック」を用いたプログラムがそのままプログラムを変更せずに動くことを可能にし、また同時にそのようなプログラムが安全に動作することを示唆している。