# 審 査 の 結 果 の 要 旨

氏　名　　大 岩　寛

C言語においては、柔軟なポインタ操作が可能であるため、プログラムに対して意図しない入力を与えることにより、メモリ上の配列の境界を越えたアクセスを行わせ、その上のデータ構造を破壊することが可能である。このような欠点により、C言語で書かれた各種のプログラムは常にネットワークからの攻撃に晒されている。本論文は、メモリへの意図しないアクセスを許さないC言語の実装手法を提案し、実際に実装した処理系の効率に関する評価を行っている。

第1章では、本論文の背景と目的について詳述してある。Java、C#、Lisp、ML等の現代的な言語においては、メモリへのアクセスに対して保護機構が用意されているが、C言語にはそのような機構はない。これは、C言語がアセンブラ言語を代替する高級言語として柔軟で直接的なメモリ操作を特徴としていたことと、C言語が設計された30年前にはこのような保護機構を導入することが現実的でなかったためである。その後、計算機環境は激変したが、C言語で書かれた既存のプログラムは多く存在し、C言語やそのプログラミングスタイルに慣れ親しんだプログラマも数多い。本論文では、メモリに対する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、しかも実行効率の十分高いC言語の処理系 Fail-Safe C を実装することを目標としている。

第2章においては、Fail-Safe C の基本的な概念について解説してある。すなわち、ポインタの表現方法(fat pointer)、型の情報を含むメモリブロック、メモリ管理、構造体と共用体の実現方法、関数の表現方法について解説し、Fail-Safe C の設計に関する基本的な考え方を説明してある。

第3章においては、実装した Fail-Safe C 処理系における各種の技法に関して詳述してある。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している。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している。「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。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ている。

第4章では、実装した処理系の効率を、いくつかの小規模なプログラムと、BYTEmarkのベンチマークに対して評価している。通常のC言語の実装と比較して、プログラムによっては数十%のオーバーヘッド、悪くても数倍のオーバーヘッドで実行できることが示されている。また、実在する有名なプログラムに存在するセキュリティ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。

第5章では、本論文の貢献をまとめ、C++への拡張など、今後の課題について述べている。

本論文は、以上に述べたように、Fail-Safe C 処理系の設計と実装により、C言語の新しい実装技術を開発したものであり、ネットワークにおけるセキュリティ技術としても貢献が大きい。よって、本論文は博士（情報理工学）の学位請求論文として合格と認められる。