5ZA-02

コンパイラがプログラムに対して与える安全性の定量的評価方法の提案とGCCを使用した実験 *

坂本健士郎 †　山口文彦 ‡

長崎県立大学情報システム学部情報セキュリティ学科 §

1 はじめに

近年、サイバー攻撃による被害が増大し、セキュリティを求められる傾向が強まる中で、プログラミングにおいてもセキュリティが求められるようになった。しかし、セキュリティを意識してソースコードを書く場合には開発コストがかかってしまう。開発コストを出来るだけ抑えるために、安全な実行コードを出力するコンパイラの開発が進められてきた。そのようなコンパイラにVITC[1]やFail-Safe C[2]がある。安全性の評価としてFail-Safe Cでは実用的で脆弱であることが知られた2つのソフトウェアをコンパイルし、攻撃に対してエラー検出のレポートが実行時に出力できることを確認している。コンパイラが安全な実行コードを出力してくれることをいうのであれば上記のような評価だけでいいが、安全なコードを出力する能力はコンパイラの性能の一つであって、コンパイラ間で比較できるような定量的な指標があると良いと考えた。指標を得るためには定量的評価方法も必要になる。定量的に安全性を評価することで安全な実行コードを出力するコンパイラの能力の比較が可能になり、安全な実行コードを出力するコンパイラの普及に繋がると考えられる。そこで、本研究ではCWE[3]が規定する複数の脆弱性を持つソースコードを用意し、攻撃が回避された割合を安全性提供度とする定量的評価方法を提案する。さらに提案する方法に従ってGCCを評価する実験を行った。

2 安全性の定量的評価方法

本研究で提案する安全性の定量的評価方法を以下に記述する。この評価方法は複数の脆弱性を持つソースコードに対してどの程度安全性を与えているのか定量的に評価するために定義した。

1. 複数の脆弱性のそれぞれに対して次の1.1と1.2を実施する。

1.1. 対象の脆弱性に対して安全性を与えるオプションを選定し、選定したオプションを使ってコンパイルする。

1.2. プログラムに対して攻撃ができるかどうか検証する。

2. 検証した結果を以下のように定量的に評価する

$$\text{安全性提供度} = \frac{\text{攻撃が回避された脆弱性の数}}{\text{実装した脆弱性の数}}$$

プログラムへの攻撃を行う際のOSの設定として、スタック保護機能、データ実行防止機能、アドレス空間配置のランダム化の機能を無効にする。

また、GCCを使用した実験においてオプションを警告オプション[4]とコード生成オプション[5]と最適化オプション[6]に絞って採用している。しかし、警告オプションは脆弱な部分を指摘するだけで実際には安全なコードを出力するわけではない。すなわち攻撃の回避には二種類あって、一つは実行時に攻撃を防ぎ止めること、もう一つは攻撃の原因である部分を表示して、危険であることを警告することである。安全性提供度はこの二種類の和であると考えて、GCCでの実験の場合は次のように評価する。

安全性提供度=静的安全性提供度 + 動的安全性提供度

---

* A criterion of compilers' safety optimization by benchmarks and its experiment on GCC

† Kenshirou Sakamoto

‡ Fumihiko Yamaguchi

§ Department of Information Security, Faculty of Information Systems,University of Nagasaki

$$\text{静的安全性提供度} = \frac{\text{警告を出された脆弱性の数}}{\text{実装した脆弱性の数}}$$

$$\text{動的安全性提供度} = \frac{\text{攻撃を防止された脆弱性の数}}{\text{実装した脆弱性の数}}$$

3 結果

実装した脆弱性は 42 種類、採用したオプションは 34 種類で実験を行った。これらの全組み合わせ 1428 通りのうち攻撃を回避のは 60 通りであった。安全性提供度としては次のような値となった。

$$\text{静的安全性提供度} = \frac{10}{42}$$

$$\text{動的安全性提供度} = \frac{9}{42}$$

$$\text{安全性提供度} = \frac{19}{42}$$

実験の結果により今回用意した脆弱性ベンチマークプログラムに対して約 4 割の安全性を与えていることが分かった。

4 考察

評価実験を行う中で、次の問題点が見つかった。

・脆弱性を突く攻撃として一種類しか検証しない場合、他の攻撃が成功するかどうかを評価できていない

・攻撃の頻度に差があるものに対しての安全性を同じように評価してしまう

上記の問題点を考慮して安全性評価方法の改善案を次に述べる。

・脆弱性を持つプログラムは脆弱性に対する攻撃を全て行えるように実装する。

・全ての攻撃を攻撃の頻度に合わせて数段階に分けて点数化する。

・安全性提供度は次のように算出する。

$$\text{安全性提供度} = \sum_{i=1}^{\text{オプション数}} \text{オプションの安全性提供度}_i$$

$$\text{オプションの安全性提供度} = \text{攻撃ごとの点数} \times \text{悪用の成否 (悪用成功なら 0、失敗なら 1)}$$

この方法は脆弱性に対して可能な攻撃を全て行い、攻撃の頻度によって評価の差別化を図るのでより厳密に安全性提供度を評価出来ると考えられる。この場合、各攻撃にどのように点数付けするかが課題となる。

5 まとめと今後の課題

本研究ではコンパイラがプログラムに与える安全性を定量的に評価する方法を提案し、GCC を評価する実験を行った。GCC を使った実験ではプログラムに約 4 割の安全性を与えていることが分かった。今後の課題として、安全性提供度は相対評価なので多くのコンパイラに対して提案手法を実施することが必要である。また、評価方法の実施、特に攻撃が成功するか否かの検証に時間がかかるため、自動評価を行えるようにすることが挙げられる。さらに、OS などの実行環境の設定を踏まえた上での評価方法が必要である。

参考文献

[1] 古瀬 淳,”VITC:情報流解析による高安全コンパイラ”, 情報処理学会第 70 回全国大会, 講演論文集 vol5,p389–p390, 2008

[2] Oiwa Yutaka,”Implementation of a Fail-Safe ANSI C Compiler”,The University of Tokyo, 2004,Ph.D. thesis.

[3] CWE-658:2019.CWE-658:Weaknesses in Software Written in C

[4] ”Warning options-Using the GNU Compiler Collection (GCC)”.3.8 Options to Request or Suppress Warnings,https://gcc.gnu.org/onlinedocs/gcc-5.4.0/gcc/Warning-Options.html#Warning-Options,(accessed 2020-01-09)

[5] ”Code Gen Options – Using the GNU Compiler Collection (GCC)”.3.18 Options for Code Generation Conversions,https://gcc.gnu.org/onlinedocs/gcc-5.4.0/gcc/Code-Gen-Options.html#Code-Gen-Options,(accessed 2020-01-09)

[6] ”Optimize Options - Using the GNU Compiler Collection (GCC)”.3.10 Options That Control Optimization,https://gcc.gnu.org/onlinedocs/gcc-5.4.0/gcc/Optimize-Options.html#Optimize-Options,(accessed 2020-01-09)