メモリ破損脆弱性を利用した攻撃に対するRustを用いた対策手法
Countermeasure method using Rust against attack using memory corruption vulnerability

千脇 貴之・システム分科会・情報セキュリティ大学院大学

In existing system programs, memory corruption vulnerability is a problem, and the Rust language is attracting attention as a countermeasure. In this research, Rust language is connected to existing programs at low cost, and the damage by the attack using the memory corruption vulnerability is reduced. In the proposed method, we consider replacing a part of the library written in C language with a library written in Rust language. As a result, it was confirmed that the memory corruption vulnerability which occurs in C language can be prevented in the library in which the memory operation is completed in the function..

現状

- システムプログラムでは、プログラマのミスが、境界外へのアクセスやダングリングポインタを引き起こす
- C/C++言語では、メモリ破損脆弱性がプログラム全体で発生する可能性がある

プログラマがメモリ管理の責任を全て負うメモリ破損脆弱性を利用した攻撃が混入し得るプログラム領域が広い

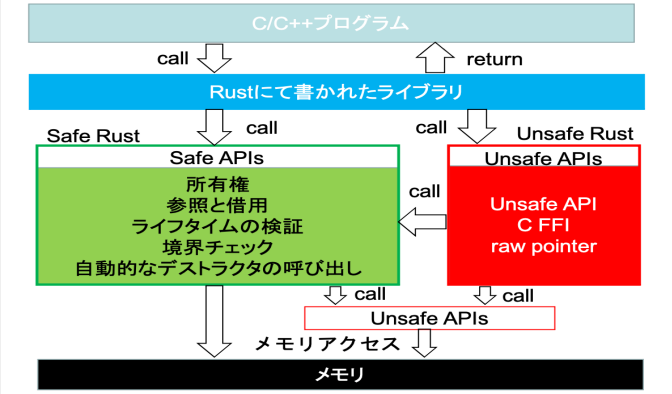
目的

C言語のメモリ操作をRustに置き換え、安全性を高める

C言語での開発において、現在のプログラムやコンパイラに大きな変更を加えない安全性確保手法を検討

手法

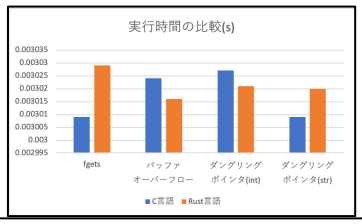
C言語のライブラリをRust言語に置き換える



実装

- fgets()(標準入力)
  - Cから渡される情報に依存する
- アロケータ関数
  - freeが必要なため安全にするのは難しい
- 配列境界外アクセス
  - 境界アクセスチェックがありエラーとなる
- バッファオーバーフロー
  - 入力値分を動的確保するため起きない
- ダングリングポインター
  - 所有権とライフタイムにより値が破棄される

評価



実行速度比較

ライブラリ関数	C 言語のサイズ	Rust 言語のサイズ
fgets	None	241KB
境界外アクセス	12KB	171KB
バッファオーバーフロー	12KB	207KB
ダングリングポインタ	12KB	261KB

ディスク容量比較

まとめ

- メモリ確保領域の情報がC言語からの情報に依存するため、渡される情報が間違っていると、メモリ破損脆弱性が発生する
- ライブラリ内でメモリ操作が完結する場合には、関数を呼び出すようにリンクを変更することで、C言語で起こるメモリ脆弱性を防ぐことが可能
- 大きなオーバヘッドが発生しない
- コストが小さく、プログラマがリンクを変更することで、Rust言語の安全性を適用可能