Return-Oriented Programmingによる難読化手法の検証

金沢工業大学 工学部 情報工学科4年　1800799　嶋田壮志

研究背景

現代では多くのソフトウェアが世に排出され様々な分野で活かされている。
しかしその一方で悪意の持った人がソフトウェアを解析し、作成者が意図してない改造や不正に情報を取得することが行われている。

例）ゲームのチート行為、有料ソフトウェアの不正利用

このような不正を防ぐ手法の一つに**「ソフトウェア難読化」**がある。

ソフトウェア難読化の手法と目的

- プログラムを意図的に読みづらくする
- プログラムに余計な処理を含ませる

➡ 解析に手間を取らせるように工夫すること

ソフトウェア難読化の例

無意味な計算

```
c = a + b
↓
c = (a*2 + b*2)/2
```

無意味な条件分岐(pnは命令列)

```
p1...
p2...
↓
if( (必ず真を取る複雑な条件式) ){
    p1...
    p2...
} else {
    p3...
}
```

提案手法

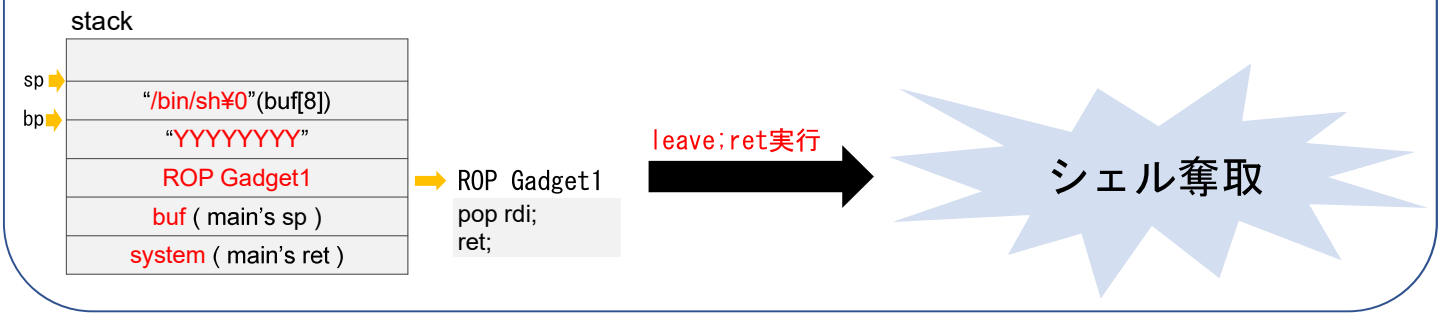
攻撃で使われる技術である「Return-Oriented Programming（ROP）」を使用した
ソフトウェア難読化手法の実装と評価

「Return-Oriented Programming（ROP）」

スタックと呼ばれるメモリ領域上に実行したいプログラムのポインタを連続して配置し、順次実行する攻撃手法。

ROPの例（シェルコード）

スタック上をこのように書き換え、leave;ret命令を実行したときシェル奪取が行われる



本研究では、通常のプログラムをROPの動作のようにスタックベースで実行するようにバイナリを書き換えることで、プログラムの実行フローを複雑にし、バイナリからプログラムを推測する静的解析を防ぐことを目的とする。

評価指標

- Completeness（完全性）
  この方法で難読化のCode Coverage（コード網羅率）
- Performance（性能）
  この難読化がパフォーマンスに与える影響
- Correctness（正確さ）
  プログラムのセマンティクスが保たれているか

今後の活動予定

- ROP難読化プログラムの作成
  逆アセンブラ、バイナリ書き換えの実装
- 評価指標に則った評価
  一般的なOSSに難読化処理を施し、評価する