SecHack365

# Ela
# x86_64向けツールチェインの開発

学習駆動コース　坂井ゼミ ／ 安藤慎

## 001 概要

Elaは、x86_64向けのツールチェインを**フルスクラッチ**で開発するプロジェクトである。
開発を通してツールチェインについての理解を深め、**独自のセキュリティ機構**を実装する。
このプロジェクトで開発されたツールチェイン及びライブラリは**OSS**として公開する。

GitHub

## 002 ツールチェインとは

実行可能ファイルの生成及び実行のために利用されるソフトウェアの総称。
ここでは、コンパイラ・アセンブラ・リンカ・エミュレータを指している。

## 003 成果

### 成果1: ツールチェイン

特徴
- **フルスクラッチ**で開発
- **ELFファイル**に対応
- **Rust**を用いて既存のライブラリに依存せず実装

ソースコード

↓

**コンパイラ**
- 独自言語をコンパイル
- 中間表現を用いて最適化

↓

**アセンブラ**
- x86_64アセンブリをアセンブル
- 24命令に対応

↓

**リンカ**
- オブジェクトファイルをリンク
- スタティックリンクに対応

↓

**エミュレータ**
- ELFファイルを実行

↓

実行

### 成果2: 独自のセキュリティ機構

特徴
- ツールチェインを**連携**させて実現
- **既存のプログラム**に対しても一部適用可能

**TME = Typed Memory Entry**

メモリ上の値に型情報を付加することによって、不正なメモリ操作を防ぐ。

特徴
- 独自セクションに型情報を埋め込む
- 実行時にメモリ操作を検証する
- メモリ操作をより細かく制限できる

**SFE = Signed Function Entry**

関数に署名の情報をつけることで、実行時に実行可能ファイルの改ざんを検査し不正なコードの実行を防ぐ。

特徴
- コンパイル時に関数の署名情報を埋め込む
- エミュレータが署名情報を用いて検証を行う

### 実行例

```
user01 at arch in ~/s/g/p/ela (master)
↪ cat ./src/read_write.vd
func read(fd: int, buf: *byte, count: int): int
func write(fd: int, buf: *byte, count: int): int

func main(): int {
  var buf: byte[128]
  read(0, buf, 128)
  write(1, buf, 128)
  return 0
}
```

ソースコード

ツールチェインを用いて実行 →

```
user01 at arch in ~/s/g/p/ela (master)
↪ make run FILE=read_write.vd
cat ./src/stdlib.vd ./src/read_write.vd > ./tmp/read_write.
vd
./bin/sigrun  ./tmp/read_write.vd ./tmp/tmp.s
./bin/rota ./tmp/tmp.s ./tmp/tmp.o
./bin/rota ./src/crt0.s ./tmp/crt0.o
./bin/herja ./tmp/crt0.o ./tmp/tmp.o ./a.out
./bin/eir ./a.out
hogehoge
hogehoge
Exited with code 0
```

実行結果

## 004 今後の展望

- ツールチェインから得られる情報を収集・整理し、生成された実行可能ファイルの解析に利用する
- より多くの命令やアーキテクチャに対応する
- 既存のツールチェインとの連携を実装する

NICT 国立研究開発法人 情報通信研究機構