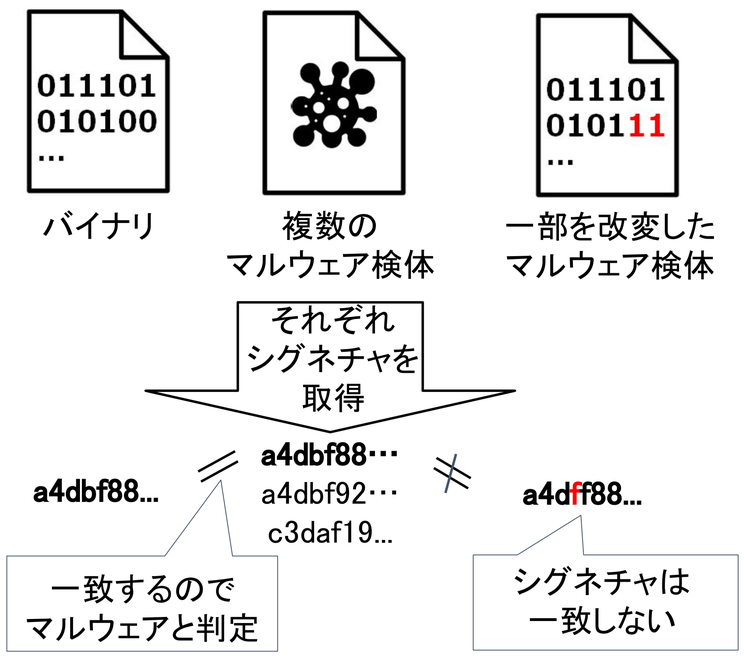
SecHack365

MalGraph :CFGを用いたマルウェア検知システム

研究駆動コース 美馬隆志

シグネチャを用いた検知手法の限界



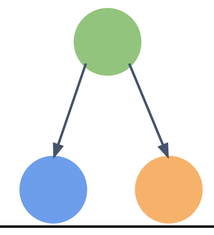
中間表現として優れたCFG

問題点 / 着目部分	検知した根拠を提示できる？	検知にかかる時間は短い？
バイナリ[1]	X	O
ふるまい[2]	O	X
CFG (Control Flow Graph)	O	O

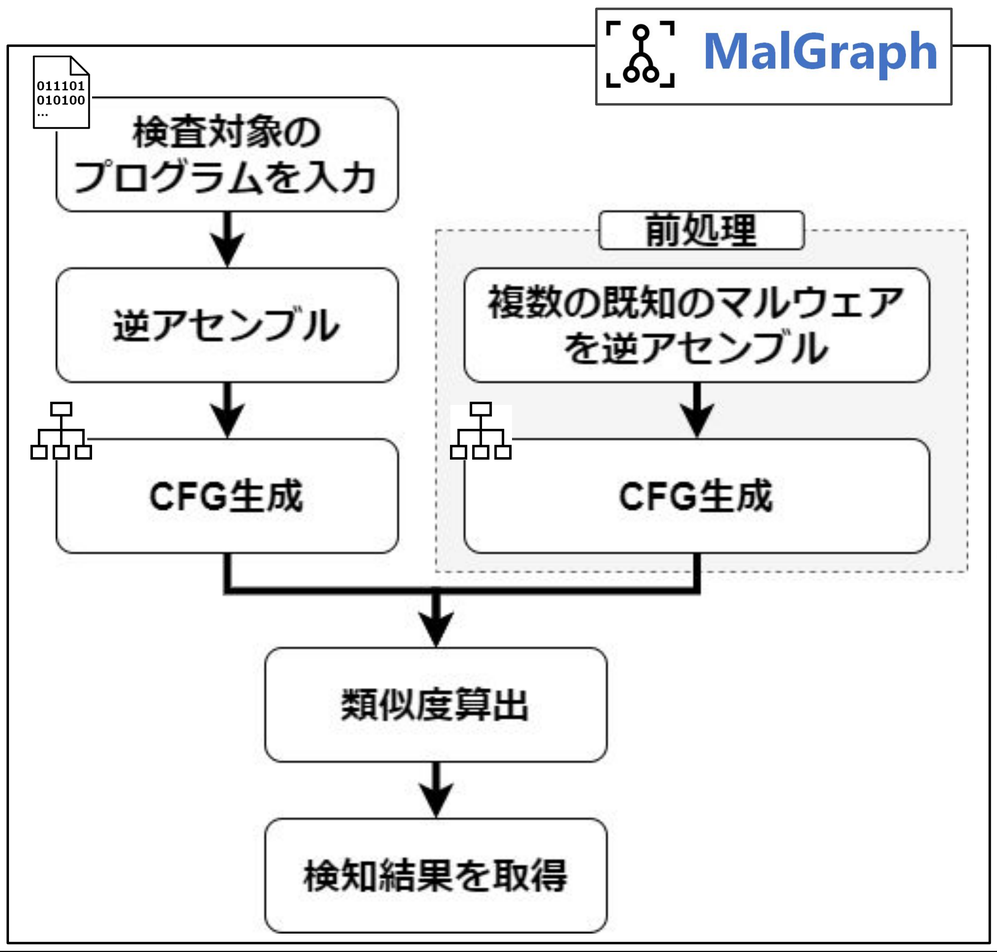
- 実行時に通る可能性のある全経路を基本ブロック単位でグラフ化したもの
- 些細な改変に影響されないデータ構造

```
if (a>b) {
  printf("a>b");
} else {
  printf("b≧a");
}
```

CFGを生成



提案:CFGを中間表現としたマルウェア検知システム



既知のマルウェアの一部を改変した未知のマルウェアでも検知可能

SecHack365ではこちらを実装

CFG生成部

- 複数の既知のマルウェア検体と検査対象のプログラムをCFGに変換する
- 強制的にフラグレジスタを書き換えることで、プログラムの全経路を網羅的に探索

類似度算出部

- CFG生成部で得られたCFG同士の類似度を算出する
- グラフ埋め込みを用いてグラフをベクトル化し、教師あり学習を用いて識別器を学習
- 学習させた識別器を用いて類似度を二値化して、検知に利用する

CFG生成部の実装と評価実験

- 強制的にフラグレジスタを書き換えることで、ファイルを網羅的に探索
- 小規模なサンプルプログラムとbzip2、無作為に選択したマルウェア検体で実行時間およびメモリ使用量を計測

		サイズ(KB)	実行時間(s)	メモリ使用量(KB)
分岐命令	32bit	7.2	1.13	6.7
	64bit	8.3	1.34	6.5
無限ループ	32bit	7.2	1.06	5.9
	64bit	8.3	1.07	5.9
bzip2	32bit	34.9	exit()システムコールでCFG生成が強制終了したため計測不可	
マルウェア検体	32bit	48.2		

32bitと64bitで実行時間に変化は見られない

評価実験結果の考察

- 小規模のファイルは網羅的に探索できる
- CFGは32bit/64bitに関係なく生成できる
- CFG生成が強制終了する問題は対処できる
  - +解析前におけるexit()システムコールのnop埋め
  - +解析時におけるexit()システムコールスキップ

→短時間でCFGを生成することができ、提案システムの実行時間の短縮に繋がる

今後の展望

- グラフ類似度算出部の実装
- 強制終了を意識したCFG生成部の実装改善

NICT 国立研究開発法人 情報通信研究機構 GitHub: github.com/task4233/malgrapher

[1] Kim et al., "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders", *Information Science 2018*.
[2] Wang et al., "Heterogeneous Graph Matching Networks for Unknown Malware Detection", *IEEE Big Data 2019*.