6K-05

# シンボリック実行を利用した 動的ソフトウェアバースマークの抽出システムの検討

松田 隼汰 † 神崎 雄一郎 † 光本 智洋 † 玉田 春昭 ‡

† 熊本高等専門学校 ‡ 京都産業大学

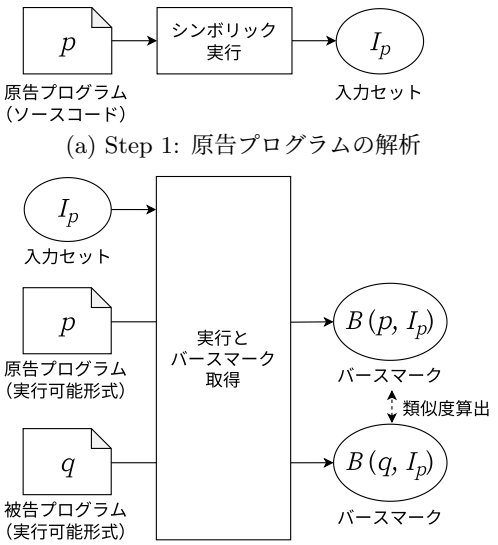
## 1 はじめに

ソフトウェアの盗用，すなわち，ソフトウェアを構成するプログラムをライセンスに違反して利用する行為を発見する手段として，ソフトウェアバースマークの技術が提案されている [1]．ソフトウェアバースマーク（以下，単にバースマークと呼ぶ）とは，プログラムから抽出可能な，ソフトウェアの機能を特徴付ける情報のことであり，メソッドの呼出し履歴 [1] や命令列（オペコード列）の k-gram [2] など，様々な種類が提案されている．バースマークによる盗用判定では，まず原告プログラム（身元が明らかなプログラム）と被告プログラム（盗用か否かの判定対象のプログラム）からそれぞれバースマークを抽出する．そして，それらを比較し類似度を算出する．この類似度により盗用であるか否かを判定する．

バースマークのうち，動的バースマークは，プログラムを実行させることで得られる情報をもとにしたバースマークであり，プログラム全体の盗用を検出するのに適しているとされる [3]．動的バースマークは，プログラムの変形に対して比較的堅牢である一方で，プログラムの構造を理解し，多くのパスを通るように実行させるための入力セットを準備する必要がある．そのため，バースマークの抽出に大きな労力を要する傾向がある [3]．そこで本研究では，プログラムの入力セットを自動的に取得できるシンボリック実行を利用して，動的バースマークを効率良く抽出するためのシステムの構築を目指す．

## 2 提案システム

シンボリック実行を用いた動的バースマークの抽出方法として，原告プログラムと被告プログラム両方に対してシンボリック実行を行い，得られた入力セットに基づいてそれぞれのバースマークを得る方法が考えられる．しかし，被告プログラムが難読化されている場合には，シンボリック実行の適用が困難な場合がある．また，被告プログラムが実行可能形式でしか入手できない場合は，ソースコードを対象にしたシンボリック実行ツールの適用ができないため，使用できるシンボリック実行ツールが少なくなる．そこで提案システムでは，ソースコード形式の原告プログラムに対してのみシンボリック実行を行い，それによって得られた入力セットを，実行可能形式の原告プログラムと被告プログラム両方に与えてそれぞれのバースマークを得るという方法をとる．




(a) Step 1: 原告プログラムの解析

(b) Step 2: 実行とバースマークの取得

図 1 提案システムの概略

提案システムの概略を図 1 に示す．図 1 では，原告プログラムと被告プログラムがそれぞれ 1 つずつの場合について示している．提案システムは，原告プログラムの解析 (Step 1) と，バースマークの取得 (Step 2) の 2 段階によってバースマークの抽出処理を行う．まず Step 1 では，原告プログラム $p$ のソースコードに対してシンボリック実行を行い，$p$ についての入力セット $I_p$ を取得する．続く Step 2 においては，Step 1 で得られた入力セット $I_p$ を，原告プログラム $p$ および被告プログラム $q$（どちらも実行可能形式）に与えて実行し，それぞれバースマーク $B(p, I_p)$ および $B(q, I_p)$ を取得する．$B(p, I_p)$ と $B(q, I_p)$ は，それぞれ $I_p$ に含まれる入力の数だけ実行時情報を含むことになる．そのため，$I_p$ 内の同じ入力に対する実行時情報の間で類似度を算出し，それらの平均値や最大値を $B(p, I_p)$ と

**A System for Extracting Dynamic Software Birthmarks Using Symbolic Execution**

Hayata Matsuda†, Yuichiro Kanzaki†, Tomohiro Mitsumoto†, Haruaki Tamada‡

†National Institute of Technology, Kumamoto College

‡Kyoto Sangyo University

表 1　原告プログラムの分析結果

| 対象 | 入力数 | 実行トレース長 最小 | 最大 |
|---|---|---|---|
| fact | 36 | 118 | 198 |
| rand | 2 | 241 | 242 |
| col | 100 | 95 | 2,327 |

表 2　バースマーク間の類似度

| 原告プログラム | 被告プログラム fact | rand | col | fact_main | fact_enc | fact_rec |
|---|---|---|---|---|---|---|
| fact | 1.00 | 0.220 | 0.505 | 0.687 | 0.621 | 0.447 |
| rand | 0.265 | 1.00 | 0.203 | 0.153 | 0.207 | 0.150 |
| col | 0.465 | 0.130 | 1.00 | 0.358 | 0.289 | 0.251 |

$B(q, I_p)$ の間の類似度とする．この値により，$q$ の $p$ の盗用に関する疑いの有無を判定する．

## 3　実験

### 3.1　概要

提案システムによって動的バースマークが期待どおりに抽出できるかを確認するための実験を行う．本実験における原告プログラムは，素因数分解を行う fact，コラッツ予想の計算を行う col，入力をシード値とした乱数を生成して特定の数値と比較する rand の 3 つとする．これらは，文献 [4] に記載されたものをもとに作成した C 言語のプログラムである．また，被告プログラムは，上述の原告プログラムに加え，fact の処理をすべて main 関数に配置した fact_main，fact に対して Tigress[1] による Encode Arithmetic の難読化を適用した fact_enc，fact と同様の素因数分解の処理を再帰を用いて記述した fact_rec といった計 6 つのプログラムとする．被告プログラムのコンパイルには，GCC (バージョン 7.5.0) を用いる．

シンボリック実行は，動的シンボリック実行ツールである KLEE[2]（バージョン 2.3）を用いる．なお，原告のソースコードに対しては，KLEE による解析のために，プログラム中の scanf 関数を用いた入力処理を klee_make_symbolic 関数を用いた表現に変換する，入力値の範囲（本実験では 0 以上 100 以下とする）を指定する命令を挿入する，といった変更を加える．抽出するバースマークは，各入力に対する実行トレース（実行されたアセンブリ命令列）のオペコードの 3-gram の出現頻度を要素とするベクトルとし，バースマーク間の類似度は，同じ入力に対するベクトル間のコサイン類似度の平均値とする．なお，実行トレースは，angr[3]（バージョン 9.2.6）を用いて取得し，命令以外のコードが命令として解釈されたと推測される部分は除外する．

### 3.2　結果と考察

KLEE によるシンボリック実行によって得られた各原告プログラムの入力 (テストケース) の数，および，各入力を自身のプログラムに与えて実行した場合に得られた実行トレース長（命令数）の最小値と最大値を表 1 に示す．KLEE によって得られた入力の数は，プログラムによって様々であった．rand の入力数が少なく，実行トレース長の変化が小さいのは，一定回数の繰返しと 1 つの条件分岐で構成される単純なプログラム構造であるためと考えられる．一方，入力値が繰返しの条件に関係する col については，入力値の範囲内のほぼすべての値が入力セットに含まれており，実行トレース長も差が大きい．なお，得られた入力セットに対するソースコードの行単位でのカバレッジを gcov を用いて計測したところ，fact と rand が 100%であったのに対し，col は約 85%であった．

表 2 に示すのは，各バースマーク間の類似度である．fact のソースコードをもとに作成した fact_main および fact_enc は，他の 3 つのプログラムと比べて fact との類似度が高い．また，プログラムの目的が異なる fact，rand，col の間の類似度は低い場合が多いが，fact と col 間の類似度は比較的高くなっている．

実験を通して，2 章で述べた手順でバースマークの抽出・比較が行えることを確認できた．

## 4　おわりに

本研究では，シンボリック実行を利用して動的バースマークをプログラムから抽出するシステムを検討した．実験では，試作したシステムを用いてバースマークを抽出できることを確認した．3.1 で述べたように，KLEE の解析に必要な変更をプログラムに加える必要はあるものの，その後の処理については自動化できるため，動的バースマークを効率良く抽出できるといえる．一方，シンボリック実行によって入力セットを求める処理が，プログラムによっては十分に機能しない場合があった．この問題への対処方法の検討や，規模を拡大した実験の実施が，今後の課題である．

**謝辞**　本研究の一部は，JSPS 科研費 JP20K11761 および JP19K11916 の助成を受けた．

## 参考文献

[1] H. Tamada, M. Nakamura, and A. Monden, “Design and evaluation of birthmarks for detecting theft of Java programs,” IASTED Inter. Conf. Software Engineering, pp.569–574, 2004.

[2] G. Myles and C. Collberg, “K-gram based software birthmarks,” Proc. 2005 ACM Symposium on Applied Computing, pp.314–318, 2005.

[3] 横井昂典，玉田春昭，“単体テストコードとアスペクト指向を用いた動的バースマークの抽出コストの削減,” 情報処理学会論文誌，vol.60，no.7，pp.1247–1259，July 2019.

[4] 奥村晴彦，C 言語による標準アルゴリズム事典，技術評論社，2018.

[1] Tigress: https://tigress.wtf/
[2] KLEE: https://klee.github.io/
[3] angr: https://angr.io/