7K-06

難読化変形の機械学習による判別の困難さに関する実験的評価

北岡 哲哉 † 神崎 雄一郎 † 森川 みどり ‡ 門田 暁人 *

† 熊本高等専門学校 ‡ (株) ワイズ・リーディング * 岡山大学

1 はじめに

プログラムコードの難読化は，ソフトウェアに対する不正な解析・改ざん行為を困難にする方法として広く用いられている [1]．難読化されたコードに対する攻撃の 1 つに，逆難読化 (難読化前の状態に近づけるためのコード変形) がある．攻撃者が逆難読化を試みる際，コードにどのような難読化が適用されているかという情報が手がかりになり得るため [2]，逆難読化を防ぎたい防御者にとっては，難読化変形 (難読化のためにコードに適用された変形内容) の判別の困難さを把握することは重要となる．そこで本研究では，難読化変形の判別の困難さに関する実験的評価を行う．具体的には，演算表現の複雑化や制御構造の平滑化といったよく知られた難読化変形について，コードの各断片の出現頻度を特徴とした教師あり機械学習による多クラス分類を試み，分類性能の評価指標を用いて難読化変形の自動判別の困難さを議論する．

先行研究として，著者らは，コードのステルス評価を目的に，コードが難読化されているかどうかをランダムフォレストで判定 (2 クラス分類) する実験を行っている [3]．また，Salem らは，命令の TF-IDF 値を用いた機械学習によって難読化変形を予測する方法を提案している [2]．本研究では，異なるアプローチとして，命令の N-gram の出現頻度を特徴とした機械学習によって難読化変形を多クラス分類する実験を行い，(1) どのような難読化変形が判別困難か，(2) コードのどのような性質を捉えて難読化変形が判別されるか，という点を中心に考察する．

2 判別の困難さの評価方法

本研究では，1 章で述べたように，難読化変形の判別の困難さを，機械学習によって構築した多クラス分類モデルを用いて評価する．具体的な評価の流れを図 1 に示す．まず，難読化されていないコード群と，それを難読化したコード群を，難読化変形ごとに分けて用意する．次に，各コードについて，コードを構成するアセンブリ命令の N-gram (例えば，3-gram であれば `push - mov - sub` など) の出現頻度，すなわち，出現回数をコードの長さで正規化したもの (出現率) を求める．これらの出現頻度のデータ (各値が出現率で表された bag of N-grams) を訓練データとした教師あり機械学習によって，各難読化変形 (難読化なしを含む) を予測する多クラス分類モデルを構築する．構築したモデルにテスト用のコード群を与えたときの分類性能の評価指標 (例えば，正解率や F1 値) に基づいて，各難読化変形の判別の困難さを議論する．

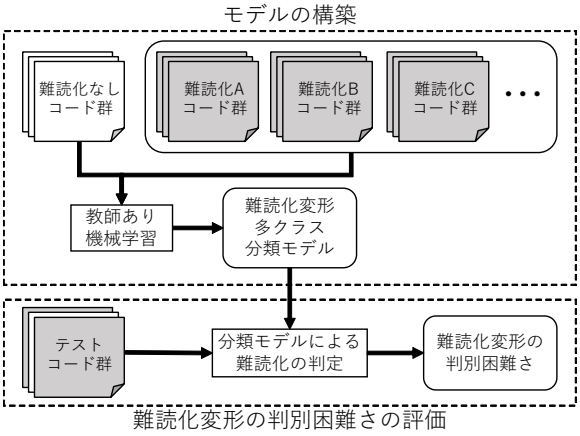

図 1 難読化変形の判別困難さ評価の流れ

3 ケーススタディ

難読化変形の判別困難さ評価の流れ (図 1) に沿った実験を行い，結果について議論する．評価対象の難読化変形は，演算表現の複雑化 (以降，EncA)，制御構造の平滑化 (以降，Flat)，5 個の opaque predicate の挿入 (以降，Opaq) とした．難読化対象のコードとしては，著者らの先行研究 [3] と同様，文献 [4] に掲載されている 261 個の C 言語プログラムから得られる 1,262 個の関数を用いた．難読化は，上述の難読化方法が実装された難読化ツール Tigress[1] を用いて行った．各コード

[1] Tigress: `http://tigress.cs.arizona.edu/`

A Case Study on the Automatic Detection of Obfuscating Transformations via Machine Learning

Tetsuya Kitaoka†, Yuichiro Kanzaki†, Midori Morikawa‡, Akito Monden*

†National Institute of Technology, Kumamoto College

‡Y's Reading Inc.

*Okayama University

表 1　各 N-gram についての分類モデルの正解率

学習アルゴリズム	N-gram の長さ		
	1-gram	2-gram	3-gram
ランダムフォレスト	0.951	0.966	0.956
決定木	0.858	0.847	0.829

表 2　決定木 (1-gram) の分類性能

難読化変形	再現率	F1 値
難読化なし	0.927	0.813
EncA	0.636	0.733
Flat	0.867	0.886
Opaq	0.994	0.994

は GCC によってコンパイル後，IDA[2]によって逆アセンブルした．各コードを構成するアセンブリ命令 (オペコードのみ) の N-gram の出現頻度のデータセットを生成し，その 7 割を訓練用データとし，残りの 3 割をテスト用データとした．機械学習のアルゴリズムとしては，ランダムフォレストおよび決定木 (どちらも木の深さは 4) を用いた．なお，分類モデルの構築や分類性能の評価値の取得には，scikit-learn[3]を用いた．

構築した多クラス分類モデルの正解率 (overall accuracy) を，表 1 に示す．結果から，決定木よりもランダムフォレストの方が全体的に高い正解率を得ていることがわかる．また，2-gram 以上は命令の順序が考慮されるため，単純な命令の出現頻度である 1-gram よりも高い判別精度となることを期待したが，今回の実験の範囲内では，N-gram の長さ (N) は，正解率に大きな影響を与えていないといえる．

次に，判別方法が単純で議論しやすい 1-gram の決定木に着目し，難読化変形ごとの分類性能や分類過程について考察する．1-gram の決定木における各難読化

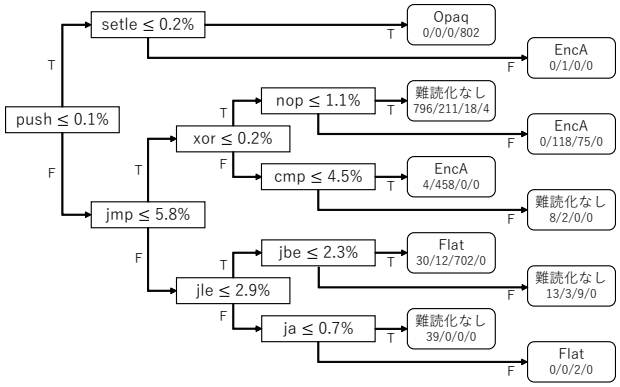

図 2　決定木 (1-gram) による難読化変形の分類

変形の再現率 (recall) および F1 値 (F1 score) を表 2 に示す．表 2 から，EncA は再現率，F1 値ともに比較的低いことがわかる．データを確認すると，EncA のコードを難読化なしと誤って予測されていることが多かった．一方，Flat や Opaq のコードの多くは正しく分類されていた．従って，EncA は今回の対象の中では判別困難さが比較的高い難読化変形であるといえる．続いて，1-gram の決定木のグラフを図 2 に示す．終端ノードには，判定クラスと各クラスのサンプル数 (難読化なし/EncA/Flat/Opaq) を示している．図 2 から，`jmp` 命令の出現頻度が高い場合は Flat に分類されやすいことや，`xor` 命令，`nop` 命令，`cmp` 命令の出現頻度を用いて，難読化なしか EncA かを判定していることなどがわかる．なお，Opaq の多くは `push` 命令の出現頻度で判別できたが，これは opaque predicate の挿入によってコードが長くなることで `push` 命令の出現率が下がったためだと考えられる．

4　おわりに

本研究では，難読化変形の判別の困難さを，機械学習によって構築した多クラス分類モデルを用いて評価・分析する実験を行った．結果から，演算表現の複雑化によって難読化されたコードは，難読化されていないコードと判別がつきにくい場合があることなどがわかった．また，決定木の分類の振舞いから，制御構造の平滑化では `jmp` 命令が多用されるといったような難読化変形の性質を分析できた．今後の課題として，難読化変形の種類の数やデータの規模を大きくした場合についてのケーススタディを行うことが挙げられる．

謝辞

本研究は，JSPS 科研費 JP19K11916 の助成を受けたものである．

参考文献

[1] C. Collberg and J. Nagra, Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Program Protection, Addison-Wesley Professional, 2009.

[2] A. Salem and S. Banescu, “Metadata recovery from obfuscated programs using machine learning,” Proceedings of the 6th Workshop on Software Security, Protection, and Reverse Engineering, pp.1:1–1:11, 2016.

[3] 北岡哲哉，神崎雄一郎，森川みどり，門田暁人，“ランダムフォレストを用いた難読化されたコードのステルス評価の検討,” 第 18 回情報科学技術フォーラム (FIT2019) 講演論文集，pp.29–32，Sept. 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.

[2]IDA: `https://www.hex-rays.com/products/ida/`
[3]scikit-learn: `https://scikit-learn.org/`