5J-07

実行トレースに着目したソフトウェア保護機構のステルス考察

松田 篤和 †
神崎 雄一郎 ‡
† 熊本高等専門学校 電子情報システム工学専攻
‡ 熊本高等専門学校 人間情報システム工学科

1 はじめに

ソフトウェアに含まれる秘密情報 (ライセンスチェック機構など) を攻撃者から保護するために，従来，数多くのプログラム難読化方法が提案されている [1]．デバッガなどの動的解析のツールを用いて取得できる難読化されたプログラムの実行トレース (実行された命令の履歴) には，実行命令数が明らかに多い，極めて出現頻度の低い命令列が出現する，というような難読化特有の特徴が現れる場合がある．このような特徴は，難読化されているコード (および保護している秘密情報) の位置を特定する手がかりを攻撃者に与え，結果として保護の強さを大きく低下させる可能性があるため，その特徴を把握するための方法が求められる．そこで本研究では，実行トレースに現れる難読化の特徴の度合いを評価するための指標として，実行トレースに出現する命令数，命令の種類数，および，不自然なトレース断片数の 3 つを提案する．提案する指標は，難読化されたコードの動的解析に対するステルス (発見されにくさ) の評価の一助となると考える．ケーススタディでは，既存の 3 つの難読化方法によって難読化された各コードのステルスについて，提案する指標に基づいて考察する．

2 実行トレースとトレース断片の不自然さ

ある入力を与えてプログラムを実行したとき，プログラムの開始から終了までに実行された命令を実行順に並べたものを実行トレースと定義する．実行トレース中の各命令は，アセンブリ命令 (Intel 表記) のオペコードとオペランドの種類の組で表す．オペランドの種類は，IDA によるオペランド分類 [2] (例えば，0 はオペランドなし，1 はレジスタ，4 はディスプレースメントを伴うメモリ参照，5 は即値を示す) を用いる．例えば，アセンブリ命令 “`mov eax, [ebp-10h]`” は `mov14`，“`push ebp`” は `push10` と表される．また，実行トレース中の一定数の連続する命令をトレース断片と呼ぶ．

トレース断片の不自然さは，その断片がアセンブリの命令列としてもっともらしくない度合を表すものである．文献 [3] では，n-gram モデルを用いてアセンブリ命令列の不自然さを評価する方法が提案されている．本稿では，トレース断片が数命令の長さであることを前提とし，命令 $i_1, i_2, \ldots, i_n$ で構成される長さ n のトレース断片の不自然さを，簡易的に $-\log P(i_n|i_1^{n-1})$ として求める．ここで，i_n はトレース断片の n 番目の命令，i_1^{n-1} はトレース断片の 1 番目から $n-1$ 番目までの命令列を示す．命令列 i_p^q がコーパスに出現する頻度を $c(i_p^q)$ とすると，上記条件付確率 $P(i_n|i_1^{n-1})$ は，最尤推定を用いて $c(i_1^n)/c(i_1^{n-1})$ と計算できる (ゼロ頻度問題を避けるために，確率値のスムージングが行われる)．

3 難読化の特徴評価の指標

実行トレースに現れる難読化の特徴の度合を評価するための指標として，「命令数」，「命令の種類数」および「不自然なトレース断片数」を提案する．「命令数」および「命令の種類数」は，それぞれ実行トレースに出現する命令の総数，および，ユニークな命令数 (重複する命令を取り除いた命令数) を示す．また，「不自然なトレース断片数」は，実行トレースを n-gram 単位に分割することで得られる各トレース断片のうち，与えられたしきい値を超える不自然さを持つトレース断片の数を示す．本稿のケーススタディでは，しきい値を $\mu + 3\sigma$ とする．ここで μ および σ は，コーパスに含まれる全 n-gram のアセンブリコード断片の不自然さの平均値および標準偏差をそれぞれ示す．

4 ケーススタディ

提案指標を用いて，既存方法によって難読化されたコードのステルスを考察する．このケーススタディにおけるトレース断片の不自然さを評価するためのコーパスは，Cygwin* を構成する 3,071 個の実行可能ファイル (Windows PE 形式) の `.text` セクションを逆アセンブ

```
int checklicense(void) {
  time_t current_time;
  int code;
  time(&current_time); /* set current time */
  if (current_time > mktime(&EXPIRE_TIME)) {
    printf("Enter activation code: ");
    scanf("%d", &code);
    if (code == ACTIVATION_CODE) {
        renewlicense();
    } else {
     printf("Wrong code.\n");
        return -1; /* failure */
     }
   }
   return 0; /* success */
}
```

図 1 オリジナルプログラムの一部

A Study on the Stealth of Protected Code Based on Its Execution Traces
Atsuto Matsuda†, Yuichiro Kanzaki‡
†Advanced Course of Electronics and Information System Engineering, National Institute of Technology, Kumamoto College
‡Department of Human-Oriented Information Systems Engineering, National Institute of Technology, Kumamoto College

*Cygwin: `https://www.cygwin.com/`

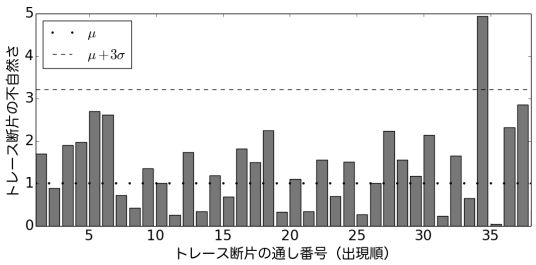
表 1　各実行トレースの分析結果

名称	命令数	命令の種類数	不自然なトレース断片数
T_o	38	18	1
T_{enca}	106	32	10
T_{camf}	42	19	3
T_{opaq}	124	25	2

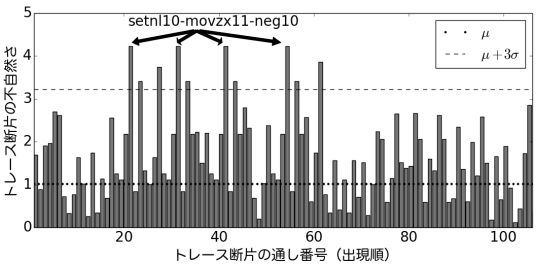
ルして得られたコードによって構築した．難読化対象となるプログラムの C 言語のソースコード (一部) を図 1 に示す．このプログラムは，ライセンスの期限をチェックし，期限が切れていた場合，ユーザにアクティベーションコードの入力を求める．ここでは，ライセンスの期限が切れている状況で，攻撃者がアクティベーションコードをランダムに入力するという場面を想定し，その場合における実行トレースを取得した．難読化前のプログラムの実行トレースを T_o と表す．このプログラムを，定数 `ACTIVATION_CODE` を扱う演算の表現の複雑化 [1]，比較命令など 4 つの命令のカムフラージュ [4]，10 個の opaque predicate [1] の挿入の 3 通りの方法で難読化した．各方法で難読化されたプログラムの実行トレースを，それぞれ T_{enca}，T_{camf}，T_{opaq} と表す．

表 1 に各実行トレースに含まれる命令数，命令の種類数，不自然なトレース断片数を示す．また，3-gram 単位で分割した T_o および T_{enca} のトレース断片の不自然さの値を図 2 に示す．縦軸はトレース断片の不自然さを示し，トレース断片の出現順に並べている．コーパスの 3-gram の不自然さの平均値 μ および不自然かどうかを判定するしきい値 $\mu + 3\sigma$ も示している．

表 1 より，T_{enca} と T_{opaq} の命令数と命令の種類数は，T_o と比較して大幅に増加していることがわかる．この特徴は，攻撃者にコードが難読化されいてることを気づかせる手がかりを与え，難読化されたコードのステルスを低下させる可能性があると考えられる．また，不自然なトレース断片数は T_{enca}，T_{camf}，T_{opaq} のいずれも増加している．特に大きく増加した T_{enca} に新たに生じた不自然なトレース断片は，全て難読化されたコードの一部であった．図 2(b) に示すように，特に不自然さの大きい 4 つのトレース断片はいずれも `setnl10-movzx11-neg10` である．整数演算の複雑化に用いられる `setnl10` と `neg10` の組合せなどが，不自然なトレース断片を多く生じさせたと考えられる．また，T_{camf} に新たに生じた不自然なトレース断片は，全てカムフラージュされた命令を実行時に元来の命令に復帰させる命令を含むトレース断片であった．この命令が挿入されたことによりプログラムの意味的なつながりが破壊され，不自然なトレース断片が生じたと考えられる．以上に示した不自然なトレース断片は，いずれも難読化されたコードの一部であり，攻撃者に難読化されたコードの位置を特定する手がかりを与え，難読化されたコードのステルスを低下させる要因になると考えられる．T_{opaq} に新たに生じた不自然なトレース断片は opaque predicate の初期化処理の一部であった．しかし，これは `checklicense` 関数を呼び出す `main` 関数に存在する単純な構造体の代入処理であり，このトレース断片が難読化されたコードの位置を特定する手がかりを与えてステルスを低下させるとは考えにくい．



(a) T_o を構成する各 3-gram の不自然さ



(b) T_{enca} を構成する各 3-gram の不自然さ

図 2　T_o および T_{enca} のトレース断片 (3-gram)

5　おわりに

本研究では，実行トレースに現れる難読化の特徴の度合いを評価するための指標として，実行トレースに出現する命令数，命令の種類数，および，不自然なトレース断片数の 3 つを提案し，既存方法によって難読化されたコードのステルスを考察した．結果から，難読化によって命令数やその種類の数，不自然な (めずらしい) トレース断片の数が大きく変化する場合があることがわかった．そのため，提案指標は難読化されたコードの動的解析に対するステルス評価の一助となると考える．

参考文献

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

[2] Hex-Rays, IDA Support, `https://www.hex-rays.com/products/ida/support/`

[3] 神崎雄一郎，尾上栄浩，門田暁人：コードの「不自然さ」に基づくソフトウェア保護のステルシネス評価，情報処理学会論文誌 Vol.55，No.2，pp.1005-1015，February 2014.

[4] 神崎雄一郎，門田暁人，中村匡秀，松本健一：命令のカムフラージュによるソフトウェア保護方法，電子情報通信学会論文誌，Vol.J87-A，No.6，pp.755-767，June 2004.