ソフトウェア保護機構を構成するコードの特徴評価の試み

神崎 雄一郎 †　　門田 暁人 ‡

† 熊本高等専門学校 人間情報システム工学科
861-1102 熊本県合志市須屋 2659-2
kanzaki@kumamoto-nct.ac.jp

‡ 奈良先端科学技術大学院大学 情報科学研究科
630-0192 奈良県生駒市高山町 8916-5
akito-m@is.naist.jp

あらまし ソフトウェア保護機構を構成するコード，すなわち，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ためにソフトウェアに追加されたコードの特徴の大きさを評価する方法を提案する．ここでは，文書に含まれる単語の特徴を重み付ける方法である TF-IDF を用いて，プログラム中のコード (命令列) の特徴の大きさを定量的に評価する．提案方法を用いれば，攻撃者にソフトウェア保護機構の発見・除去の大きな手がかりを与えるといわれるステルシネスの低いコードを効率的に検出し，ソフトウェア保護機構の信頼性の評価に役立てることができる．

Evaluating the Characteristics of Software Protection Code

Yuichiro Kanzaki†　　Akito Monden‡

†Dept. of Human-Oriented Information Systems Engineering,
Kumamoto National College of Technology
2659-2 Suya, Koshi, Kumamoto 861-1102, Japan
kanzaki@kumamoto-nct.ac.jp

‡Graduate School of Information Science,
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-0192 Japan
akito-m@is.naist.jp

Abstract This paper presents a method for evaluating the characteristics of software protection code, which has been added to the original program to protect it against malicious reverse engineering attacks. We exploit the TF-IDF weighting scheme, often used in the information retrieval domain, as a measurement tool to quantify the characteristic of instructions (sequence) in a program. The proposed method helps to efficiently detect unstealthy code, in which adversaries might easily identify and destroy its protection mechanisms.

1　はじめに

不正な解析や改ざんを防止するためにソフトウェアに追加された保護機構には,「特徴的」なコード，すなわち，一般的に出現頻度が低い命令や命令構成を持ったコードが含まれる場合が多い．例えば，実行時間を利用した改ざん防止方法 [3] や動的解析防止方法 [7] を用いた保護機構は，一般的には出現頻度の低い，タイムスタンプカウンタを取得する命令 (RDTSC 命令 [6]

など) を含む．また，暗号技術を利用した保護機構 [2] には，本来のプログラム内容にかかわらず暗号化や復号のための特有の命令列が含まれる．Collberg らが [3] で述べているように，このような特徴的なコードは，保護機構のステルシネス (露見のしにくさ) を弱め，攻撃者に保護機構の発見・除去の大きな手がかりを与えることになる．そのため，ソフトウェア保護を適用する際には，保護のために追加・変更したコードの特徴の大きさを把握し，それを最小限に保つことが重要となる．そこで本研究では，ソフトウェア保護機構を構成するコードの特徴を定量的に評価する方法を提案する．具体的には，情報検索の分野で索引語の重みづけ方法として用いられる TF-IDF 法によってコードの特徴の大きさを評価する．提案方法を用いれば，ステルシネスの低いコードを効率的に検出し，ソフトウェア保護機構の信頼性の評価に役立てることができる．

以降，2 章では，プログラム，および，特徴評価の指標として用いる TF-IDF の定義について述べる．3 章では，提案方法のケーススタディについて報告する．4 章では関連研究について述べ，最後に 5 章において，まとめと今後の課題を述べる．

2 コードの特徴評価

2.1 プログラムの定義

まず，プログラムに関する定義を行う．1 つのソフトウェアを構成する**命令**の列を**プログラム**と呼び，p で表す．命令は i で表し，1 つのオペコードと 0 個以上のオペランドを持つアセンブリ形式とする．n 個の命令を持つプログラムは，$p = (i_1, i_2, \ldots, i_n)$ となる．プログラムの集合は P で表す．また，プログラム p に含まれる，要素が連続する任意の長さの部分列を**コード**と呼び，c で表す．p の x 番目の要素を先頭とする長さ j のコード c は，$c = (i_x, i_{x+1}, \ldots, i_{x+j-1})$ となる．

2.2 特徴評価の指標

本研究では，コードの特徴の大きさを評価するために，自然言語の文書における索引語 (文書の内容を特徴付ける単語) の重みづけの指標 [9] として提案されている **TF-IDF** を用いる．TF-IDF は，1 文書中の単語の頻度を表す **TF**(term frequency; 単語出現頻度) と，文書集合における単語の分布の偏りの度合を表す **IDF**(inverse document frequency; 逆文書頻度) を掛け合わせたものであり，直観的にいえば,「文書内に高い頻度で出現するが，他の文書では出現しないことが多い」単語は，TF-IDF の値が大きくなる．ここでは，TF および IDF の定義における文書集合をプログラムの集合，文書をプログラム，単語をコードにそれぞれ置き換え，TF-IDF の値をコードの特徴の大きさを示す指標として考える．

プログラムが p，プログラムの集合が P のとき，p 内のコード c の TF の値 $tf(c,p)$，IDF の値 $idf(c,P)$ は，次のように表される．

$$
\begin{aligned}
tf(c,p) &= \frac{freq(c,p)}{\sum_{x \in p} freq(x,p)} \\
idf(c,P) &= log_2 \frac{|P|}{df(c)}
\end{aligned}
$$

ここで，$freq(c,p)$ は p に出現する c の数，$|P|$ は P に属する全プログラムの数，$df(c)$ は，P に属するプログラムのうち，c を 1 つ以上含むプログラムの個数を示す．これらを掛け合わせた次式で表される TF-IDF の値 $tf\text{-}idf(c,p,P)$ を，コード c の特徴の大きさとする．

$$
tf\text{-}idf(c,p,P) = tf(c,p) \times idf(c,P)
$$

「プログラム内に高い頻度で出現するが，他のプログラムでは出現しないことが多いコード」は，TF-IDF 値が大きくなり，特徴の大きいコードとみなされる．また，Collberg らは，保護のために追加・変更されたコードと元来から存在するのコードとの区別の困難さの度合を，保護機構の**ステルシネス** (stealthiness) として定義している [3] が，特徴が大きいコードはそのよ

うな区別が容易であるとみなし，ステルシネスが低いと考える．

3 ケーススタディ

3.1 概要

提案方法によるコードの特徴評価のケーススタディについて報告する．ここでは，コード c の長さを 1(1 命令) に固定している．また，実験に用いた計算機の CPU は Intel x86 系 (IA-32 [6]) であり，アセンブリ命令は Intel 形式で表記する．

プログラムの集合 P として用意したのは，Windows のコマンドライン上で動作するアプリケーション 1,000 個 (エディタ，ゲーム，コンパイラ等) のプログラムである．対象としたファイル全体には，のべ約 2,330 万個の命令，305 種類のオペコードが含まれていた．

まず，IDF を測定し，P における各命令の大域的な特徴を調べた．その後，ファイルを圧縮・解凍するフリーソフトウェアである gzip [5](バージョンは 1.4.1) に対して，[3] に紹介されている実行時間差を用いた単純なアンチデバッギングを行う保護方法をプログラムの 20 箇所に適用し，そのプログラムの TF-IDF を測定した．また，その結果から適用した保護方法の特徴とステルシネスを考察した．

3.2 IDF の測定結果

図 1 に，各命令について IDF を測定した結果を示す (スペースの都合上，一部の命令は省略している)．縦軸が IDF の値を表す．縦軸上にプロットされていないものがあるが，見やすさを考慮したものであり，横方向の大きさに意味はない．

IDF の最小値は 0(すべてのプログラムに命令が存在する)，最大値は約 9.97(1 つのプログラムにのみ命令が存在する) となった．`mov`，`add`，`cmp` などは IDF が最小の 0，すなわち，今回の実験対象内ではすべてのプログラムに含まれており，非常に普遍的な命令であることがわかる．一方で，ビット単位でセット・リセットを行う

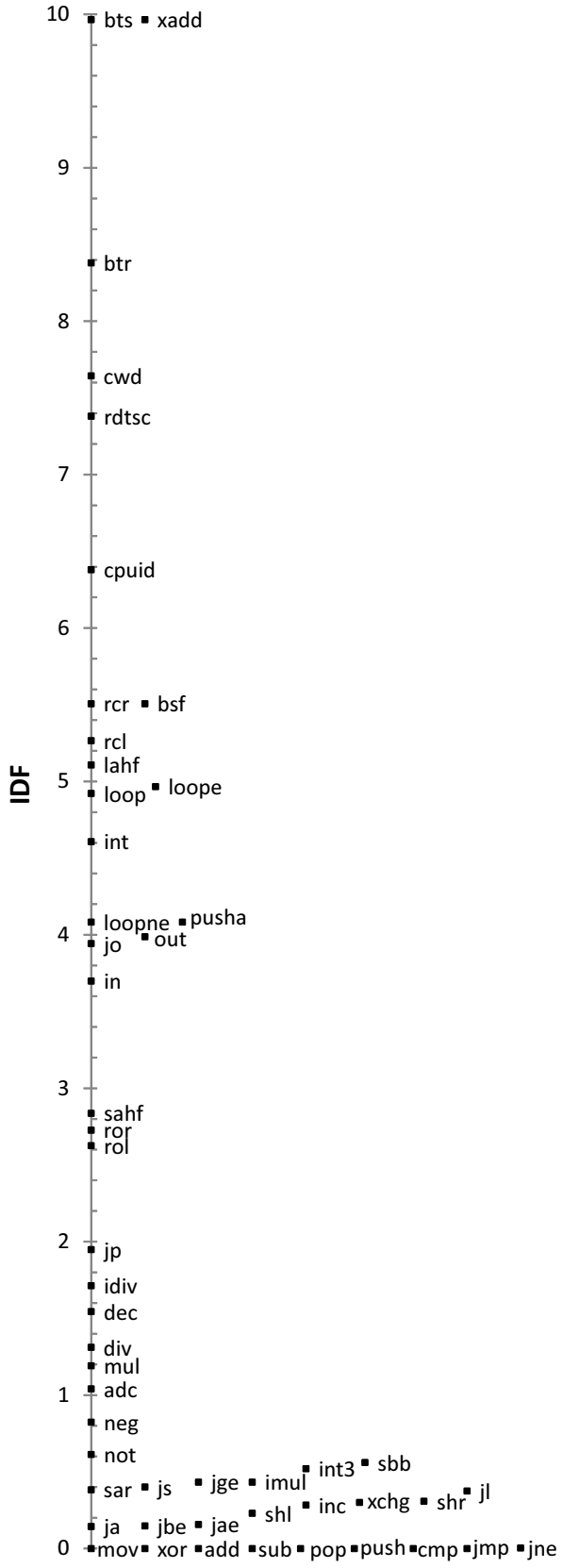
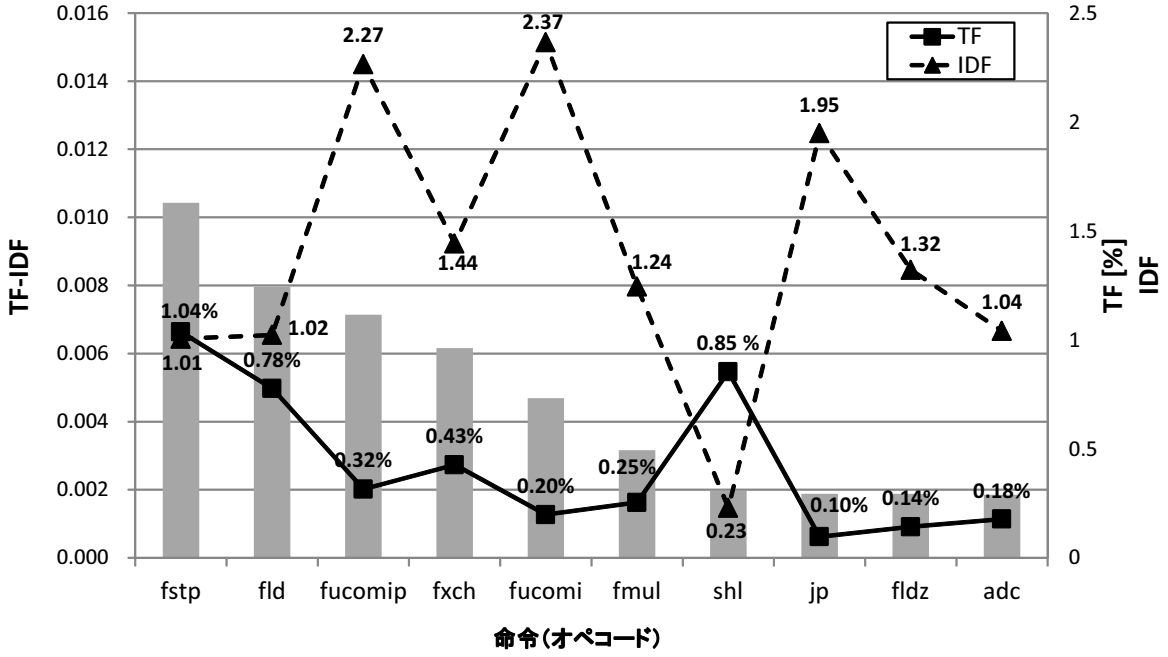
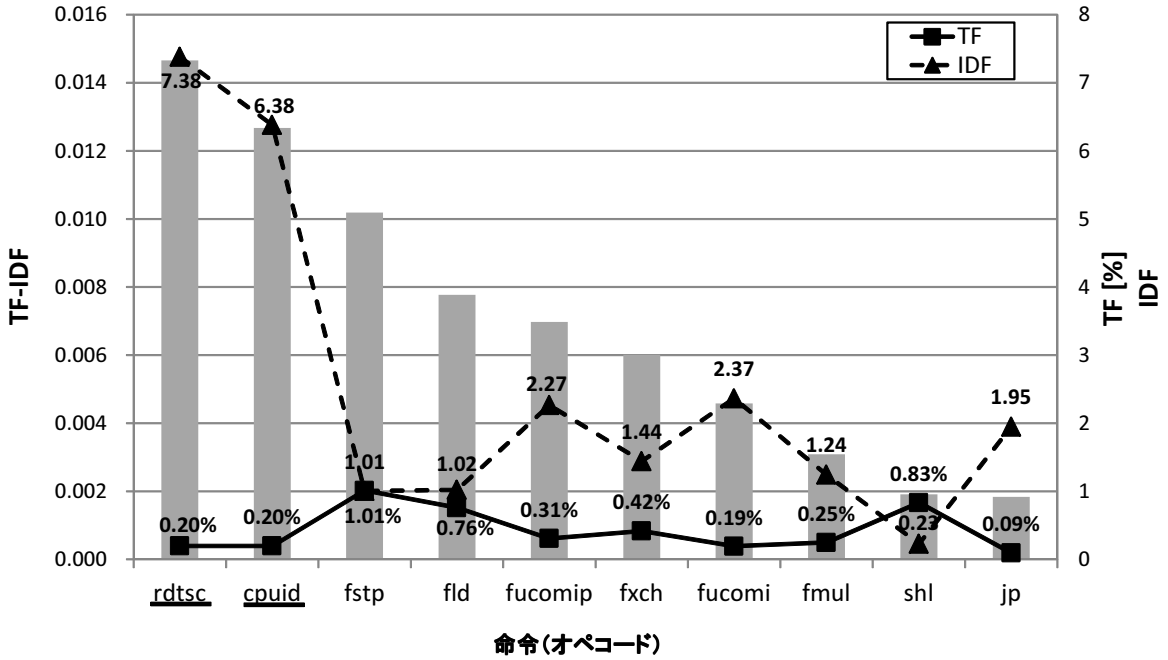


図 1: 各命令の IDF(一部抜粋)



(a) 元来の gzip のプログラムに関する TF-IDF 値



(横軸の命令のうち，保護機構に含まれる命令は下線を引いて示す．)

(b) 保護が適用された gzip のプログラムに関する TF-IDF 値

図 2: 保護の適用前後のプログラムに関する TF-IDF 値

命令 (bts や btr)，タイムスタンプカウンタを取得する rdtsc，プロセッサの情報を取得する cpuid などの命令を含むプログラムは少なく，IDF の値が高くなっている．

3.3 TF-IDF の測定結果

次に，保護が適用されたプログラムの各命令について TF-IDF を測定した結果を示し，保護機構の特徴とステルシネスについて考察する．保護が適用される前の元来の gzip のプログラムについて TF-IDF を測定した結果を図 2(a) に，また，20 箇所に保護 (アンチデバッギング) が適用された gzip のプログラムについて TF-IDF を測定した結果を図 2(b) に示す．両図とも，TF-IDF 値の大きい上位 10 命令を順に示している．ここで，横軸は命令 (オペコード) の種類，縦軸は TF-IDF の値 (棒グラフで示される)，TF の値 (単位は%，実線の折れ線グラフで示される) および IDF の値 (破線の折れ線グラフで示される) を表す．なお，図 2(b) の横軸の命令のうち，保護機構に含まれる命令は下線を引いて示す．各命令において，2 つの折れ線グラフの値 (TF および IDF) を掛け合わせると，棒グラフの値 (TF-IDF) となる．

まず，保護が適用されていない状態を対象にした図 2(a) を見てみると，fstp，fld，fucomip など，FPU レジスタを操作する命令 [6] の TF-IDF 値が大きく，対象プログラムにおいて，これらが特徴的な命令であることがわかる．fucomip，fucomi，jp などは，TF 値は fstp 等と比べて大きくないものの，IDF 値が大きいために，大きな TF-IDF 値を示している．

一方，図 2(b) を見てみると，図 2(a) に現れた命令のほかに，rdtsc と cpuid の 2 命令が大きな TF-IDF の値を示している．これらの 2 命令は，保護機構がプロセッサのタイムスタンプカウンタを用いて実行時間を測定するために追加された命令である．2 命令とも他の命令と比較して IDF 値が際立って大きいため，TF-IDF 値が大きくなっている．保護機構を構成するこれらの命令が，大きな特徴を持っており，発見しやすい (元来から存在する命令と区別しやすい) 命令になっているため，保護機構のステルシネスは低いことがわかる．保護機構のステルシネスを改善するには，プログラムの暗号化 [2] や命令のカムフラージュ [8] などを用いて特徴の大きい命令を隠す (特徴の小さい命令に置き換える) などの対策が求められることになる．

4 関連研究

ソフトウェア保護機構の性能を評価する方法として，ソフトウェアの複雑度メトリクスを用いて解析の困難さを評価する方法 [4]，メモリに現れる秘密鍵を自動検索し，発見に要する時間で保護の強さを評価する方法 [1]，命令の実行系列の差分を用いた攻撃に基づいてプログラムの耐タンパ性を評価する方法 [13] などが提案されている．提案方法は，保護機構のステルシネスについて定量的な評価を試みた点が新しい．

また，アセンブリ (機械語) レベルのコードの特徴を統計的な手法で分析するアプローチは，マルウェアを対象にした研究分野においても見られる．例として，オペコード列の出現頻度に基づいてマルウェアを効率よく検出する方法 [12]，機械語レベルでの度数分布を用いて，他のマルウェアから派生したマルウェアかどうかを判別する方法 [11]，機械語レベルで計算したエントロピー値を用いてパッキングや暗号化が適用されたマルウェアを識別する方法 [10] などが挙げられる．これらの研究がコードの特徴をマルウェアの検出や自動分類に用いているのに対し，本研究ではソフトウェア保護機構のステルシネス評価に用いている．

5 まとめ

本稿では，自然言語の文書における索引語の重みづけ方法である TF-IDF を用いて，ソフトウェア保護機構を構成するコードの特徴を評価する方法を提案した．提案方法を用いれば，攻撃者にソフトウェア保護機構の発見・除去の大きな手がかりを与えるといわれるステルシネスの低いコードを効率的に検出し，ソフトウェア保護機構の信頼性の評価に役立てることができる．実験では，公開されている 1,000 のソフト

ウェアを文書集合として用い，gzip のプログラムに適用された保護機構の特徴とステルシネスを考察した．

今後は，分析する単位となるコードの長さ，適用する保護方法，プログラムの集合 (文書集合) 等を様々に変化させて数多くの実験を行い，提案方法の信頼性を検証・改善していきたい．

参考文献

[1] 赤井健一郎，三澤学，松本勉，“ランタイムデータ探索型耐タンパー性評価法,” 情報処理学会論文誌，vol.43，no.8，pp.2447–2457，Aug. 2002.

[2] J. Cappaert, N. Kisserli, D. Schellekens, and B. Preneel, “Self-encrypting code to protect against analysis and tampering,” Proc. Benelux Workshop on Information and System Security, 2006.

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

[4] C. Collberg, C. Thomborson, and D. Low, “A taxonomy of obfuscating transformations,” Technical Report 148, Technical Report of Dept. of Computer Science, U. of Auckland, New Zealand, 1997.

[5] J. Gailly and M. Adler, “The gzip home page,” http://www.gzip.org/.

[6] インテル株式会社，IA-32 インテル・アーキテクチャ・ソフトウェア・デベロパーズ・マニュアル中巻:命令セット・リファレンス，http://www.intel.co.jp/ (Available online).

[7] Y. Kanzaki and A. Monden, “A software protection method based on time-sensitive code and self-modification mechanism,” Proc. 14th IASTED International Conference on Software Engineering and Applications (IASTED SEA 2010), pp.325–331, USA, Nov. 2010.

[8] 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.

[9] 北研二，津田和彦，獅々堀正幹，情報検索アルゴリズム，共立出版，東京，2002.

[10] R. Lyda and J. Hamrock, “Using entropy analysis to find encrypted and packed malware,” IEEE Security and Privacy, vol.5, no.2, pp.40–45, 2007.

[11] B.B. Rad and M. Masrom, “Metamorphic virus variants classification using opcode frequency histogram,” Proc. 14th WSEAS International Conference on COMPUTERS (ICCOMP 2010), pp.147–155, 2010.

[12] I. Santos, F. Brezo, J. Nieves, Y.K. Penya, B. Sanz, C. Laorden, and P.G. Bringas, Idea: Opcode-Sequence-Based Malware Detection, vol.5965 of Lecture Notes in Computer Science, pp.35–43, Springer-Verlag, 2010.

[13] 山内寛己，門田暁人，松本健一，“実行系列差分攻撃によるプログラムの耐タンパー性評価,” Information science technical report, NAIST-IS-TR2009007，Graduate School of Information Science, Nara Institute of Science and Technology，Dec. 2009.