B-037

コードの「めずらしさ」に基づく保護機構のステルシネス考察

A Study on Stealthiness of Software Protection Mechanism Based on the Rareness of the Code Fragments

尾上 栄浩 † Takahiro Onoue
神崎 雄一郎 ‡ Yuichiro Kanzaki
門田 暁人 § Akito Monden

1. はじめに

不正な解析や改ざんを防ぐためにソフトウェア保護方法によって変形・追加されたコード (以下，保護機構と呼ぶ) は，一般的なプログラムにほとんど現れない命令や命令構成を持つ場合がある．例えば，暗号化されたコードには逆アセンブリ不可能な命令や極端に出現頻度の低い命令が出現することが多い．また，命令のカムフラージュ法 [7] を用いてカムフラージュされたコードは比較命令の直後に無条件ジャンプの命令が続くなど，意味的なつながりが不自然な命令構成となる可能性がある．このような「めずらしさ」を持つコードは攻撃者に秘密情報を保護している事実や保護のメカニズムをさとられる手がかりを与えるため，保護の強さを著しく低下させてしまう恐れがある．

そこで，本研究では，コードの「めずらしさ」を定量的に評価する方法を提案し，その結果に基づいて保護機構のステルシネス [3]，すなわち保護機構が攻撃者に発見されにくい度合いについて考察する．コードの「めずらしさ」(不自然さ) の評価には IDF を用いる．具体的には，保護されたプログラムに含まれる複数の命令単位での IDF 値の測定を行い，複数命令で構成されるコードの「めずらしさ」を評価する．

ケーススタディでは，DRM を模したプログラムに Aucsmith らのプログラム暗号化法 [1][2]，神崎らの命令のカムフラージュ法 [7]，Collberg らに紹介されている実行時間差分を用いた耐タンパ法 [3] といった既存の保護方法を適用したものに対してコードの「めずらしさ」を評価し，保護機構のステルシネスを考察する．

以降，2 章では，プログラムやめずらしさに関する定義を示す．3 章では既存の保護方法が適用されたプログラムにおけるコードのめずらしさを評価する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る．最後に 5 章でまとめと今後の展望を述べる．

†熊本高等専門学校専攻科 電子情報システム工学専攻，Advanced Course of Electronics and Information Systems Engineering, Kumamoto National College of Technology
‡熊本高等専門学校 人間情報システム工学科，Dept. of Human-Oriented Information Systems Engineering, Kumamoto National College of Technology
§奈良先端科学技術大学院大学 情報科学研究科，Graduate School of Information Science, Nara Institute of Science and Technology

2. 諸定義

本研究ではプログラムに出現する命令 (列) のめずらしさを IDF を用いて評価する．まず，プログラムの定義を示し，次に IDF とそれを用いためずらしさについて定義する．

2.1. プログラムの定義

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 p は，$p = (i_1, i_2, i_3, \ldots, i_n)$ となる．プログラム p の集合は P で表す．また，プログラム p 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.IDF の定義

ソフトウェアに含まれるコードのめずらしさの評価を行うために，文書における索引語 (1 つの文書を特徴付ける単語) の重み付けに用いられる IDF(逆文書頻度) を用いる．IDF は文書集合において単語が少ない文書に偏って出現する度合いを意味し [5]，直観的には単語のめずらしさを表す指標といえる．ここでは，IDF の定義における文書集合をプログラム集合，文書をプログラム，単語をコードと置き換え，IDF 値をコードのめずらしさの度合いを示す指標として用いる．

プログラム集合を P，プログラムを p，コードを c と置いたとき，コード c のプログラム集合 P における IDF を次式のように定義する．ただし，$|P|$ はプログラム集合に含まれるプログラムの数を表し，$df(c)$ はプログラム集合 P においてコード c が 1 回以上出現するプログラムの数である．

$$idf(c, P) = log_2 \frac{|P|}{df(c)}$$

$idf(c, P)$ が大きくなるとき，コード c はプログラム集合 P 中において少数のプログラムにしか出現しないコードとなり，直観的には「めずらしい」コードとなる．本稿では，長さ (命令数) が l であるコード c のプログラム集合 P におけるめずらしさ $R_l(c, P)$ を，次のように定義する．

$$R_l(c, P) = idf(c, P)$$

Collberg らは保護のために追加・変形されたコードと元来から存在するコードとを区別する困難さの度合いをステルシネスと述べている [3]．保護のために追加されたコードや変形されたコードが大規模なプログラム集合の中でめずらしい (IDF の大きい) コードであれば，攻撃者に見つかりやすい，すなわち，元来から存在しているコードとの区別が容易であるとみなし，保護機構の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

保護されたソフトウェアを構成するコードに対して IDF(めずらしさ) を測定し，保護機構のステルシネスを考察するケーススタディを次の手順で行った．

1. プログラム集合 P として，Linux 上で動作する逆アセンブルが禁止されていない 1789 個のアプリケーションを準備し，P に出現するアセンブリコード c(c の長さは 1〜3 とする) について，$R_1(c, P)$, $R_2(c, P)$ および $R_3(c, P)$ を測定する．

   ただし，c の長さが 2 以上の場合 (R_2 および R_3 を測定する場合) については，普遍的な命令がめずらしい形で組み合わされるコードに焦点を当てて考察を行うため，測定対象を各命令の R_1 の値が全て 0.1 以下であるコードに限定する．

2. プログラム p として，DRM プレイヤを模したプログラムに保護を適用したもの (詳細は 3.2 に述べる) を用意し，p に出現するめずらしいコード (R_1, R_2 および R_3 の大きいコード) から保護機構のステルシネスを考察する．

なお，実験には CPU が Intel x86 系アーキテクチャ，OS が Ubuntu Linux である計算機を用いた．また，各アセンブリコードは実行可能ファイルを逆アセンブルすることで取得した．

3.2. 保護適用対象のプログラム

保護の適用対象として，文献 [3] で示されている DRM プレイヤのモデルを参考に実装した簡単なプログラムを用いた．図 1 に概要を示す．DRM プレイヤはユーザが持つ暗号化されたメディアを以下の手順で再生可能なメディアに変換する．

1. ユーザにアクティベーションコードの入力を促し，入力された値よりライセンスチェックを行う．
2. ユーザの持つユーザキー，DRM プレイヤに内蔵されているプレイヤキーを用いて，暗号化されたメディアを再生可能なメディアに変換する．

図 1 のプログラムに対して，既存の保護方法を適用した．適用した方法は次の 3 種類である．

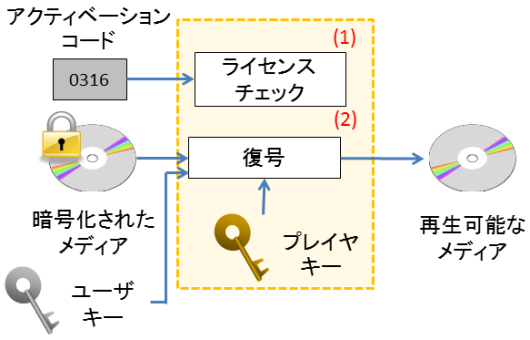


図 1: DRM プレイヤのモデル

(1) プログラム暗号化法 [1][2]
プログラム中の保護を適用したい部分を複数のセルに分割して，キーと xor 演算を用いてそれぞれのセルを暗号化する保護方法である．実行時，各セルが実行されるごとに各セルの暗号化と復号を繰り返し，結果的にどのようなタイミングにおいても保護を適用した部分は一部のセルを除き，暗号化された状態となる．

(2) 命令のカムフラージュ法 [7]
隠したい命令を予め偽の命令で書き換えておき，命令実行時の非常に短い期間のみ元の命令に復元する保護方法である．実行時，偽装された命令は実行される前に，復帰ルーチンによって元の命令に復元される．復元された命令は実行された後，隠ぺいルーチンによって偽装された命令に再隠ぺいされる．

(3) 実行時間差分を用いた耐タンパ法 [3]
コードの実行時間を利用して，デバッガ等のツールを用いた動的解析を検出する保護方法である．プログラムの一部の実行に要する時間を予想しておき，実際に実行時間を計測する．計測した実行時間が実行に要すると予想されていた時間を超えていた場合，プログラムを強制終了する．

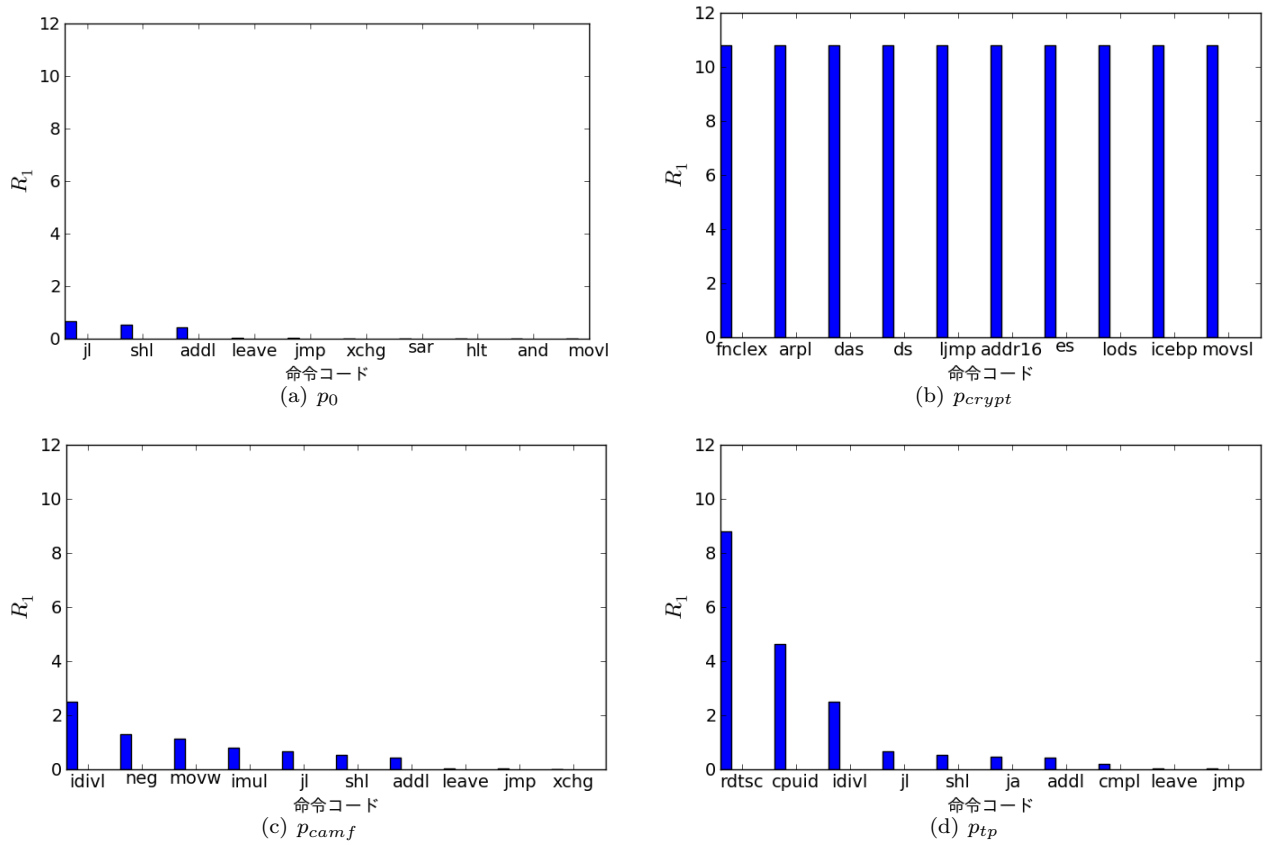
以後，便宜上，元来の DRM プレイヤのプログラムを p_0，プログラム暗号化法が適用されたプログラムを p_{crypt}，命令のカムフラージュ法が適用されたプログラムを p_{camf}，実行時間差分を用いた耐タンパ法が適用されたプログラムを p_{tp} と表す．

3.3. 実験結果と考察

p_0，p_{crypt}，p_{camf}，p_{tp} について，評価方法を用いてコード長を 1 から 3 まで変化させためずらしさ，すなわち，R_1，R_2，R_3 を測定した．各コード長ごとに評価結果からステルシネスを考察する．

3.3.1. コード長 1 としたときの評価

p_0，p_{crypt}，p_{camf}，p_{tp} から取り出したコードを R_1 が高い順に 10 個並べたグラフを図 2(a)〜図 2(d) に示す．



(a) p_0　(b) p_{crypt}　(c) p_{camf}　(d) p_{tp}

図 2: 保護アルゴリズムごとのコードの R_1 の値

図 2(a) より，保護が適用されていない元来のプログラム p_0 においては，`jl` の R_1 の値が出現した命令の中で最大であることがわかる (0.659)．

図 2(b) より，p_{crypt} においては `fnclex`，`arpl`，`das` など，R_1 が非常に高い値を示す命令が多く現れている．これらのコードはプログラム暗号化法のアルゴリズムによって暗号化された無意味なコード列の中に現れたコードである．暗号化されたコード列に滅多に使われない命令や逆アセンブリが不可能な命令などが多く現れるため，p_{crypt} には際立ってめずらしい命令が多く含まれている．p_{crypt} のプログラムには R_1 が最大の値となる命令 (プログラム集合において 1 つのプログラムにしか存在しない命令) が 29 個含まれていた．

図 2(c) より，p_{camf} では保護機構を追加するために加えた `idivl`，`movw` などの命令の R_1 が高くなっていることがわかる．この `idivl` は p_{camf} 中で最も R_1 の値が高く (2.49)，プログラム 5〜6 個に 1 つの割合で出現する．しかし，この `idivl` は目立って出現頻度が低いとはいえず，p_{camf} においては特にめずらしいといえる命令は含まれていない．

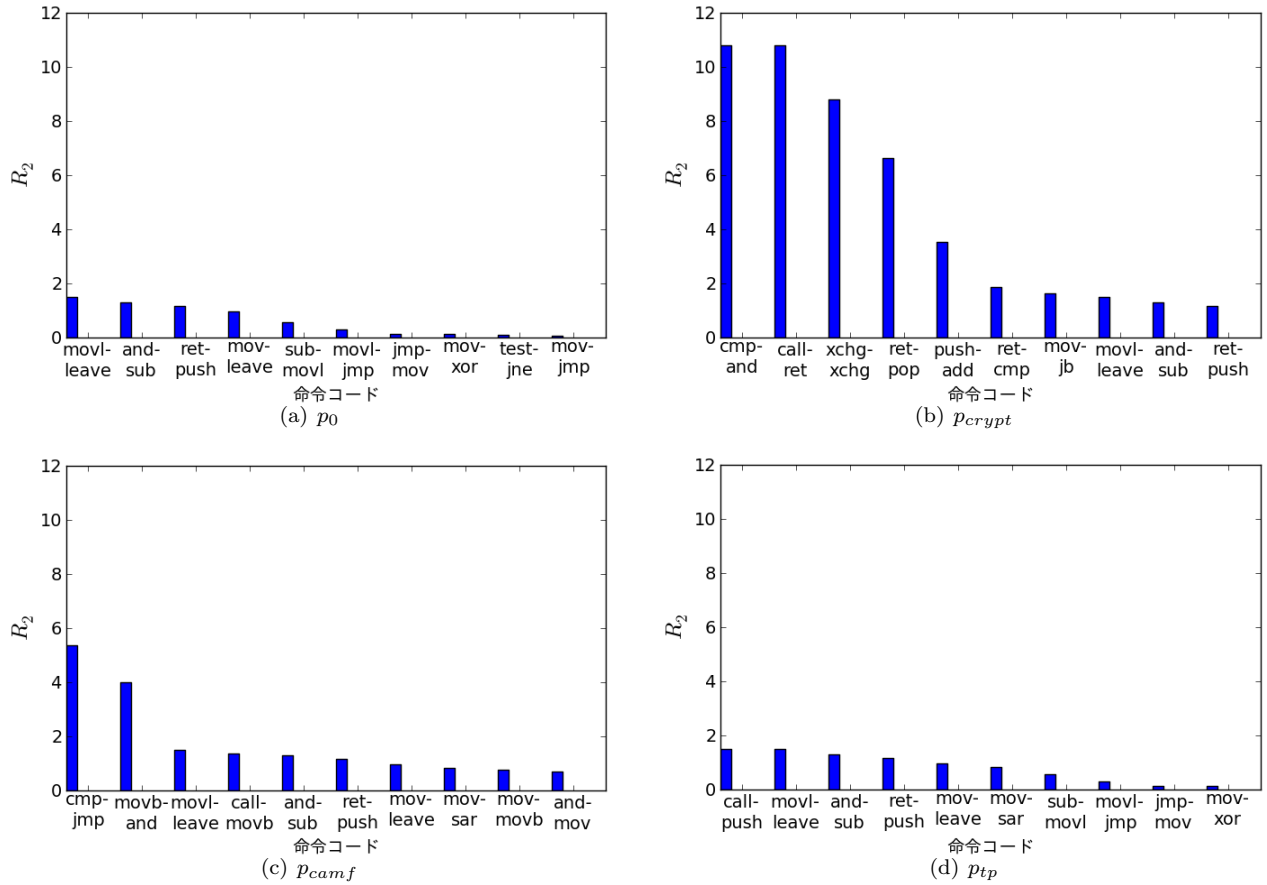
図 2(d) より，p_{tp} では `rdtsc` と `cpuid` が特に R_1 が高くなっていることがわかる．`rdtsc`，`cpuid` は実行時間を用いる保護のために用いられる命令であるが，一般的なプログラムにおいてはほとんど出現しない．従って，これらの命令は非常にめずらしい命令といえる．

以上より，コード長 1 としてコードのめずらしさを評価したとき，ステルシネスについて次のことが考察できる．プログラム暗号化法を適用したプログラムでは暗号化されたコードにめずらしい命令が多く出現し，これらの命令がステルシネスを低下させている．命令のカムフラージュ法を適用したプログラムには際立ってめずらしい命令が出現しなかったが，偽装された命令がめずらしい命令であれば，ステルシネスが低くなる可能性がある．実行時間差分を用いた耐タンパ法を適用したプログラムでは保護機構に用いられる `rdtsc`，`cpuid` 等のめずらしい命令がステルシネスを低下させている．

3.3.2. コード長 2 としたときの評価

3.3.1 節と同様にそれぞれのプログラムから取り出したコードを R_2 が高い順に 10 個並べたグラフを図 3(a)〜図 3(d) に示す．

図 3(a) より，保護が適用されていない元来のプログラム p_0 においては，`movl-leave` の R_2 の値が出現し



(a) p_0 (b) p_{crypt} (c) p_{camf} (d) p_{tp}

図 3: 保護アルゴリズムごとのコードの R_2 の値

たコードの中で最大であることがわかる (1.49).

図 3(b) より，p_{crypt} においては，`call-ret`，`xchg-xchg`，`cmp-and` のような意味的なつながりが不自然なコードの R_2 の値が高いことがわかる．プログラム暗号化法ではプログラムの一部分を無意味なコード列に暗号化しているため，このような命令の前後のつながりが壊れた命令構成となるコードが多く含まれる．これらのコードは一般的なプログラムには出現することがないため，極めてめずらしいといえる．

図 3(c) より，p_{camf} においては，`cmp-jmp`，`movb-and` のコードの R_2 の値が高いことがわかる．`cmp-jmp` は元来のコード `cmp-jne` を偽装したコードである．`cmp` はレジスタ，もしくはメモリの比較を行う命令であり，通常は `jne` や `je` などの条件付きジャンプと組み合わせて条件文として用いられる．しかし，偽装された命令を含むコード `cmp-jmp` においては比較演算を行った直後に無条件ジャンプを行う不自然な命令構成となっている．このようなコードは一般的なプログラムにはあまり見られないため，めずらしいコードといえる．

図 3(d) より，p_{tp} において，顕著に R_2 が高い値となるコードは含まれていないことがわかる．ただし，図 2(d) を見てわかるように p_{tp} には `rdtsc`，`cpuid` などの顕著に R_2 の値が高い命令が現れており，これらを含むコードの R_2 は非常に高い値を示す．しかし，3.1 節で述べているように，コード長が 2 以上の時に R_2 の値を測定するときはコードに含まれる命令の組み合わせのめずらしさに焦点を当てるため，`rdtsc` 等の R_1 が高い値となる命令を含むコードは測定対象から外されている．従って，p_{tp} に関しては命令の組み合わせという点ではめずらしいコードは含まれていないといえる．

以上よりコード長 2 としてコードのめずらしさを評価したとき，ステルシネスについて次のことが考察できる．プログラム暗号化法を適用したプログラムでは暗号化されたコード中において，命令構成が不自然となるめずらしいコードが多く出現し，これらのコードがステルシネスを低下させている．命令のカムフラージュ法を適用したプログラムでは偽装された命令が前後の命令と不自然な命令構成となるコードを構成しやすく，これらのコードがステルシネスを低下させている．実行時間差分を用いた耐タンパ法を適用したプログラムでは，不自然な命令構成となるようなめずらしいコードが出現することはなく，ステルシネスは保護による影響を受けていない．

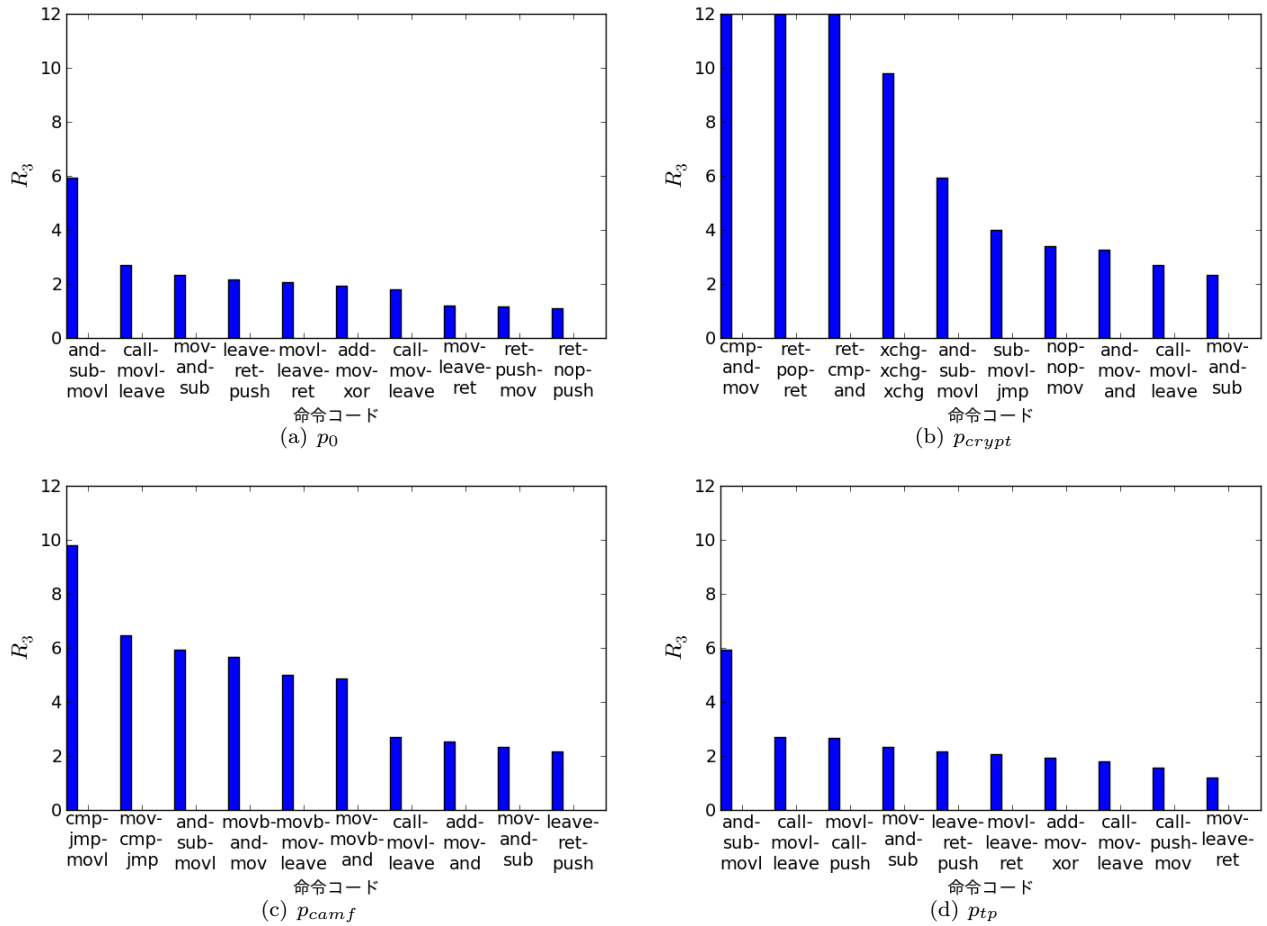


図 4: 保護アルゴリズムごとのコードの R_3 の値

3.3.3. コード長 3 としたときの評価

3.3.1 節と同様にそれぞれのプログラムから取り出したコードを R_3 が高い順に 10 個並べたグラフを図 4(a)〜図 4(d) に示す．

図 4(a) より，保護が適用されていない元来のプログラム p_0 においては，`and-sub-movl` の R_3 の値が出現したコードの中で最大であることがわかる (5.95)．

図 4(b) より，p_{crypt} においては，コード長 2 としたときにおける R_3 の値の評価と同様に，意味的なつながりが不自然なコードが際めて高い R_3 の値を示すことがわかる．特に，`cmp-and-mov`，`ret-pop-ret`，`ret-cmp-and` はプログラム集合にまったく出現せず，暗号化されたコード列にのみこれらのコードが含まれている．これらのコードは極めてめずらしいコードといえる．

図 4(c) より，p_{camf} においては，図 3(c) の `cmp-jmp` と同様に，`cmp-jmp-movl` は偽装された命令を含み，不自然な命令構成であるため，極めて高い R_3 の値を示すことがわかる．また，`movb-and-mov`，`movb-mov-leave`，`mov-movb-and` は命令の復帰，隠ぺいのために追加した保護機構を含んだコードである．これらのコードはコンパイラが生成しにくい命令の組み合わせとなっており，R_3 の値が高い．このような隠ぺいや復帰ルーチンのコードも偽装された命令を含むコードと併せてめずらしいコードといえる．

図 4(d) を見ると，p_{tp} に関しては，コード長 2 としたときにおける R_2 の値の評価と同様の原因で顕著に R_3 が高い値となるコードは出現していないことがわかる．

以上のようにコード長 3 としてコードのめずらしさを評価した場合，ステルシネスに関してはコード長 2 としてめずらしさを評価したときと同様のことが考察できる．

3.4. 議論

ソフトウェアの保護方法が適用されたプログラムにはめずらしいコードが含まれていることが多く，これらのコードがステルシネスを低くする原因となり得る．

まず，保護機構や保護されたコードに一般的には非常に出現頻度が低い命令が用いられることがある．例えば，実行時間差分を用いた耐タンパのアルゴリズムでは，保護機構において実行時間を取得するために `rdtsc`，

cpuid などの通常のプログラムではほとんど用いられない命令が追加される．また，プログラム暗号化法では実行前にプログラムの一部を無意味なコードに暗号化するため，通常のプログラムにはほとんど出現しない命令が暗号化されたコード列中に多く出現する．これらのアルゴリズムを適用したプログラムにおいてステルシネスを向上させるためには，命令のカムフラージュ法などを用いてステルシネスの低い命令を隠したり，暗号化の際に極端にめずらしい命令が出現しないようにアルゴリズムを工夫する必要がある．

ステルシネスが低いもうひとつの要因としては，保護機構や保護されたコードに，命令の組み合わせのめずらしいコードが出現することである．例えば，命令のカムフラージュのアルゴリズムによって命令を偽装した cmp-jmp のようなコードや保護のために追加したコードはアセンブリレベル，もしくはバイナリレベルで命令の追加・変更を加えられたものである．従って，これらのコードが含まれることにより，通常コンパイラが生成しない不自然な命令構成となりやすい．また，暗号化のアルゴリズムを適用したプログラムは，暗号化のため前後の命令とのつながりが考慮されないので，意味のつながりが不自然な命令構成を持つコードがプログラム中に現れやすい．これらのアルゴリズムを適用したプログラムにおいてステルシネスを向上するには，保護を適用する際にプログラムの命令のつながりが不自然にならないように，命令の前後関係を考慮する必要がある．

4. 関連研究

コードのステルシネス評価に関する方法の先行研究として，神崎らの TF-IDF 法を用いた方法がある [6]．この論文では TF-IDF(命令の頻度 TF と逆文書頻度 IDF を掛けあわせた値) を用いてコードの特徴を評価する方法を示した上で，アセンブリ一命令単位での特徴を評価する実験を行い，特徴を示した命令 (rdtsc や cpuid 等) が，ステルシネスの低下を招く危険性があることを示している．この実験では，Windows 上で動作するアプリケーション 1000 個をプログラム集合として用い，実行時間差分を用いた耐タンパ法 [3] を適用したプログラムを対象として特徴評価を行なっている．

また，統計的手法によりコードの特徴を抽出する研究として [4] がある．この論文ではマルウェアが特徴となるコード群を持つことを利用して，評価対象のプロブラムの部分列における各コードの出現頻度とマルウェアの部分列における各コードの出現頻度との類似度を測定することで，類似度の大きさからマルウェア，またはその亜種を検出することができることを論じている．その上で対象プログラムのマルウェアに対する類似度を測定する実験を行い，良性のプログラム (マルウェアでないプログラム) の類似度が低く，マルウェアの亜種の類似度が極めて高くなった結果を示している．

本研究では複数の命令単位でのコードのめずらしさを IDF を用いた測定結果から評価し，ステルシネスを考察した．

5. まとめ

複数の命令で構成されるコードにおいて，IDF をコードのめずらしさとして用いる評価方法を示し，DRM プレイヤを模したプログラムに対してプログラム暗号化法，命令のカムフラージュ法，実行時間差分を用いた耐タンパ法の 3 つのアルゴリズムを別々に適用して保護されたそれぞれのプログラムについて，保護されたコードのめずらしさを評価し，ステルシネスを考察した．

本稿ではプログラムの逆アセンブルによって得られたコードを対象にして，静的なコードに対するステルシネスの評価を行ったが，現実的には攻撃者はプログラム解析のために動的解析を用いることが主流となっている．そこで，今後の展望としてはプログラムの動的解析を想定し，プログラムの実行から得られた実行命令系列を用いてステルシネスの評価実験を試みることを考えている．

参考文献

[1] David Aucsmith. Tamper resistant software: An implementation. In *Proceedings of the First International Workshop on Information Hiding*, pp. 317–333, London, UK, 1996. Springer-Verlag.

[2] David Aucsmith and Gary Graunke. Tamper resistant methods and apparatus. US patent 5,892,899, April 1999. Assignee: Intel Corporation.

[3] Christian Collberg and Jasvir Nagra. *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection*. Addison-Wesley Professional, 2009.

[4] Igor Santos, Flix Brezo, Javier Nieves, Joseba K Penya, Borja Sanz, Carlos Laorden, and Pablo G Bringas. *Opcode-sequence-based Malware Detection*, Vol. 5965, pp. 35–43. 2010.

[5] 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, 2002.

[6] 神崎雄一郎, 門田暁人. 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. コンピュータセキュリティシンポジウム 2011 論文集, 第 2011 巻, pp. 827–832, oct 2011.

[7] 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 A, Vol. J87-A, No. 6, pp. 755–767, June 2004.