様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　２９　年　　６　月　２２　日現在

機関番号：　５７４０３
研究種目：　基盤研究(C)（一般）
研究期間：　2014～2016
課題番号：　２６３３００９４

研究課題名（和文）ソフトウェアに対するMan-At-The-End攻撃の困難さ評価

研究課題名（英文）A Method for Evaluating Software Protection Mechanisms Against Man-At-The-End Attacks

研究代表者

神崎　雄一郎（Kanzaki, Yuichiro）

熊本高等専門学校・人間情報システム工学科・准教授

研究者番号：９０４３５４８８

交付決定額（研究期間全体）：（直接経費）　　2,800,000円

研究成果の概要（和文）：本研究では，難読化等によって秘密情報が保護されたソフトウェアに対するMan-At-The-End攻撃(エンドユーザによるソフトウェアの解析・改ざん行為)の困難さ評価に関する研究を行った．特に，保護されたコードのステルス(保護されたコードがどれだけ攻撃者に発見されにくいか)の評価に重点を置いて取り組み，評価指標を複数提案するなどの成果を得た．得られた成果は，プログラムから秘密情報が保護されている箇所を探し出して理解・改ざんを試みる攻撃に対する保護の強さを正確に評価する一助になると考える．

研究成果の概要（英文）：This research proposes a method for evaluating software protection mechanisms against Man-At-The-End attacks, that is, attacks by an end user who has access to software itself and attempts to steal assets in the program. This research mainly focuses on developing metrics for evaluating the stealth of an obfuscation, that is, the degree to which obfuscated code can be distinguished from unobfuscated code. The proposed metrics help evaluate the strength of the protection against an attack which goes through a typical locate-alter-test cycle.

研究分野：　ソフトウェア保護

キーワード：　ソフトウェア保護　セキュリティ　プログラムの難読化　耐タンパソフトウェア

1. 研究開始当初の背景

ソフトウェアに対する Man-At-The-End 攻撃 (以下, MATE 攻撃と呼ぶ) とは，ソフトウェアの実行可能ファイルを物理的に所有するエンドユーザが，不正な目的のためにソフトウェアの機能を解析・改ざんする行為のことである [1]．ライセンスチェックのルーチンや DRM システムの復号鍵など，内部に秘密情報を持つソフトウェアは，MATE 攻撃を通してソフトウェアの不正な使用，コピー，改ざん等が行われる危険に常にさらされている．このような不正行為によってソフトウェアベンダが大きな損害を受けるのを防ぐため，プログラムの難読化，暗号化，耐タンパ化など，MATE 攻撃を妨げるための数多くのソフトウェア保護方法が提案されている．

保護方法を用いてソフトウェアの秘密情報の防御を試みる者にとっては，保護方法を適用することでソフトウェアがどれだけ「強く」なるか，すなわち，攻撃者が目的を達成するのに必要な労力がどれだけ大きくなるか，という点を把握することは大変重要である．一方で，保護されたソフトウェアが現実的な攻撃を受けたときに，どの程度「強い」のかを客観的に知ることは難しい．このため，保護されたソフトウェアの強さを定量的に検証するための有用な指標や方法が強く求められていた．

2. 研究の目的

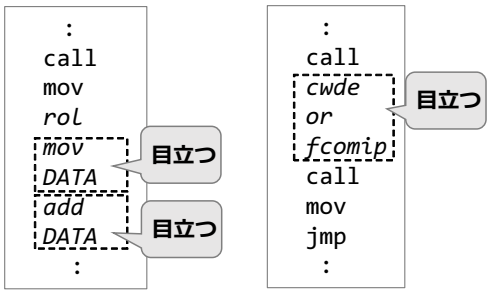
本研究の目的は，難読化等によって保護されたソフトウェアの MATE 攻撃に対する強さを評価する一方法を開発することである．特に，*locate-alter-test* の攻撃 (実行可能ファイルを逆アセンブル・逆コンパイルして得られるプログラムから，秘密情報が保護されている部分を探し出し，理解・改ざんを試みる攻撃) に対する耐性を評価するのに有効な方法を提案することを目指す．

3. 研究の方法

まず前提として，本研究では次の 2 点を満たすソフトウェアを,「保護されたソフトウェア」と考える．

(1) ソフトウェアを構成するプログラム内に，秘密情報，すなわち，攻撃者に知られたくないコードやデータ (例えばライセンスチェックのルーチンや秘密鍵のデータなど) を持つ．
(2) (1) の秘密情報の発見・理解・改ざんを困難にする目的で，プログラムの一部が難読化や暗号化などの保護方法によって変形されている．

また，保護されたソフトウェアに対して，攻撃者は典型的な *locate-alter-test* の攻撃 [2] を行う，すなわち，実行可能ファイルを逆アセンブル・逆コンパイルして得られるプログラムから秘密情報が保護されているコード (プログラムの一部) を探し出し，理解・改ざんする，という手順で攻撃を行うものと考える．



(i) 難読化されたコード　(ii) 暗号化されたコード

図 1　ステルスの低いコードの例

以上のような前提の下で MATE 攻撃の困難さを把握するには，まず，(a) 保護されているコードのステルス (stealth)，すなわち，攻撃者が保護されているコードを見つけにくい度合を評価する方法が求められる．図 1 に示すのは，ステルスが低いコードの例である．ダミー命令の挿入による難読化方法を用いて保護された (i) のコードには，コード領域にもかかわらず，逆アセンブルに失敗したコードの一部がデータ (`DATA`) となって出現している．また，暗号化によって保護された (ii) のコードには，`cwde` や `fcomip` といった極めて出現頻度の低い命令が含まれている．保護が適用されたコードにこのような攻撃者に「目立つ」部分が含まれていると，そのコードが秘密情報を保護している部分であることが容易に知られ，わずかな労力で攻撃が成功する可能性がある．そのため，ステルスの評価，具体的にはコードが攻撃者に目立つ度合を知る方法が求められる．また，秘密情報が保護されているコードが発見された後，そのコードそのものの理解がどれだけ困難か，すなわち，(b) 保護されているコードの理解に要する時間的コストの大きさを把握することも重要であるといえる．

本研究では，特に (a) のステルス評価に注目し，(1) コードが目立つ度合を測定する指標を提案した上で，(2) 目立つコードを手がかりに秘密情報を入手しようとする攻撃モデルを構築し，その攻撃の困難さを評価する実験を既存の難読化方法で保護されたプログラムを対象に行う．

4. 研究成果

(1) コードの目立つ度合を評価する指標の提案

① fragment surprisal

コードの目立つ度合を評価する指標として，まず，fragment surprisal という指標を提案した (発表論文④)．これは，アセンブリ命令で構成されるコード断片が攻撃者に目立つ度合を数値化したもので，コード断片の出現確率から得られる自己情報量をコード断片

の長さで正規化することで求める．出現確率は，多数のアセンブリプログラムで構成されるコーパスに基づいて算出される．

発表論文④での実験環境における fragment surprisal の例 (断片の長さが 3 の場合) を表 1 に示す．各ニーモニックの後に続く 2 桁の数値はオペランドの種類を表す (オペランドの種類の詳細についてはここでは省略する)．`mov-test-jz` のような非常にありふれた (出現頻度の高い) コード断片の surprisal が 2.44 である一方，`adc-adc-sal` のような非常にめずらしいコード断片の surprisal は 29.6 と高くなっている．すなわち，`adc-adc-sal` は `mov-test-jz` よりも攻撃者に目立つ度合が高いと考える．fragment surprisal の指標を用いれば，保護方法を適用して変形されたプログラムを構成するコード断片がどれだけ攻撃者に目立つかを評価できる．

表 1　fragment surprisal の例

コード断片	surprisal
`mov14-test11-jz70`	2.44
`lea14-nop00-mov31`	7.36
`adc13-adc13-sal15`	29.6

② code artificiality

また，code artificiality (コードの不自然さ) という指標を提案した (発表論文⑤)．この指標は，任意のコード全体の「不自然さ」を，N-gram モデルに基づいて算出するものである．例えば，`push`，`mov`，`test`，`jz`，`ret` という 5 命令で構成されるコードの code artificiality (3-gram の場合) は，「`<code>-push-mov`」，「`push-mov-test`」，...，「`jz-ret-</code>`」といったコード断片の出現確率を掛け合わせたものの情報量となる．ここで，`<code>`および`</code>`は，それぞれコードの先頭と終端を示す特殊な記号である．この指標によって，基本ブロック，関数，プログラム全体といった比較的大きなコード単位における「目立つ度合」を評価できる．

(2) pinpoint 攻撃に対する強さの評価

目立つコードを手がかりに秘密情報を入手しようとする攻撃モデルとして，pinpoint 攻撃と呼ぶ攻撃モデルを提案し，その攻撃の困難さを評価する実験を既存の難読化方法で保護されたプログラムを対象に行った (発表論文④)．pinpoint 攻撃は，実行可能ファイルを逆アセンブルすることによって得られるアセンブリプログラムの中で「目立つ」コード断片を探し，その部分を含むコードを理解・改ざんすることで秘密情報を得ようとする攻撃である．具体的には，アセンブリプログラムを数命令単位で順に走査して得られる各コード断片の fragment surprisal を求めた上で，一定のしきい値を超えたコード断片を目立つコード断片として検出し，それを含むコードを秘

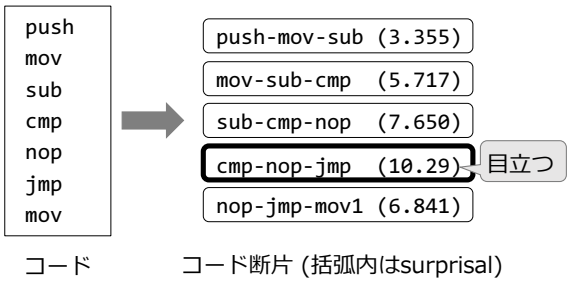


図 2　pinpoint 攻撃による目立つコード断片の検出例

密情報が保護されている場所とみなして理解・改ざんする．pinpoint 攻撃による目立つコード断片の検出例を図 2 に示す．この例では，3 命令単位でコードが走査され，fragment surprisal が 10 を超えた `cmp-nop-jmp` を攻撃者に目立つコード断片として検出している．

実験用のライセンスチェックのコードを含むプログラムを 5 種類の既存の難読化方法で保護し，それらを対象に pinpoint 攻撃を行った．その結果，opaque predicate や演算の符号化を用いた難読化を行った場合，保護されたコードの一部が目立つコード断片として検出された．図 3 は，opaque predicate を用いて難読化されたコードに検出された目立つコード断片である．このコードでは，逆アセンブルを防止するために，ランダムなバイト列がコード領域に挿入されていたが，それらがコードの文脈に合わず，目立つコード断片となっていた．目立つコード断片を含む関数には秘密情報 (ここではライセンスチェックの条件分岐) が存在し，この関数を解析できれば攻撃が成功することになるため，このコードの pinpoint 攻撃に対するステルスは低いと判断できる．一方で，制御構造の平滑化による難読化や，データの符号化による難読化が適用されたコードには，目立つコード断片が検出されなかったため，pinpoint 攻撃に対するステルスは高いと判断できる．ステル

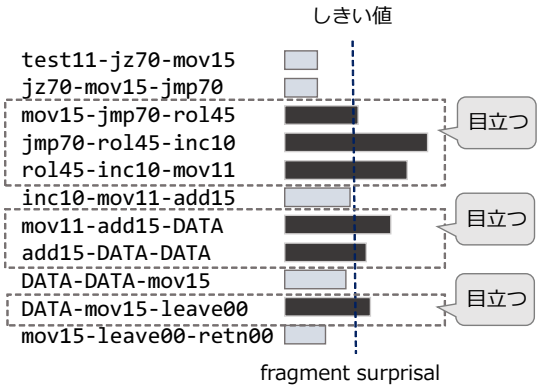


図 3　検出された目立つコード断片の例

スが低い場合，わずかな労力で前述の *locate-alter-test* による MATE 攻撃が成功する可能性があるといえる．今後の課題として，ステルス評価に加え，検出されたコードが理解しやすいかどうか，という点についても定量的に評価できれば，MATE 攻撃の困難さをより正確に評価できると考えられる．

(3) その他の成果

① 実行トレースに対するステルス評価

デバッガなどの動的解析ツールを用いて取得できるコードの実行トレースには，実行命令数が非常に多い，極めて出現頻度の低いトレース断片が出現する，というような保護方法特有の特徴が現れる場合があることに着目し，実行トレースを対象にしたステルス評価の指標を提案した (発表論文③)．具体的には，実行トレースに現れる命令の総数，命令の種類数 (ユニークな命令数)，不自然な (出現頻度が低い) トレース断片数といった指標を提案し，既存の難読化方法によって難読化された実行トレースに対するステルスを評価する実験を行った．

② 自然なコードに変形する命令のカムフラージュ法

コードの目立つ度合の指標の応用として，コードを自然な (目立たない) コードに変形する命令のカムフラージュ法 (ソフトウェア保護方法) を提案した (発表論文⑥)．提案方法では，攻撃者に隠したい命令とその周辺の命令から構成されるコードの目立つ度合を評価し，それが最小となるように変形内容を決定する．自然な命令でカムフラージュされたコードは，カムフラージュされていないコードとの区別がつきにくく，ステルスの高い保護が可能になると考える．試作システムを実装し，実験を通して提案方法の有効性について考察した．

＜引用文献＞

[1] P. Falcarin, C. Collberg, M. Atallah, and M. Jakubowski, “Software protection (guest editors’ introduction),” IEEE Software, Special Issue on Software Protection (March/April 2011), Vol. 28, No. 2, pp. 24–27, March 2011.

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

5. 主な発表論文等

(研究代表者には下線)

〔学会発表〕(計 7 件)

① 西陽太，神崎雄一郎，難読化されたプログラムに含まれる「目立つ基本ブロック」の検出，2017 年電子情報通信学会総合大会講演論文集 (講演番号 D-3-2)，2017 年 3 月 23 日，名城大学 (名古屋市)．

② 渡辺哲士，門田暁人，玉田春昭，神崎雄一郎，バイナリコード中の文字列に着目したソフトウェアの流用検出，電子情報通信学会技術報告，ソフトウェアサイエンス研究会，Vol.116，No.277 (SS2016-31)，pp.79-84，2016 年 10 月 28 日，彦根勤労福祉会館 (彦根市)．

③ 松田篤和，神崎雄一郎，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，情報処理学会第 78 回全国大会講演論文集 (講演番号 5J-07)，2016 年 3 月 11 日，慶應義塾大学 (横浜市)．

④ Yuichiro Kanzaki, Clark Thomborson, Akito Monden, and Christian Collberg, Pinpointing and Hiding Surprising Fragments in an Obfuscated Program, In Proc. of the 5th Program Protection and Reverse Engineering Workshop (PPREW-5), pp.8:1-8:9, 8 December 2015, Los Angeles (USA). (査読有)
DOI: 10.1145/2843859.2843862

⑤ Yuichiro Kanzaki, Akito Monden, and Christian Collberg, Code Artificiality: A Metric for the Code Stealth Based on an N-gram Model, In Proc. of 2015 IEEE/ACM 1st International Workshop on Software Protection (SPRO2015), pp. 31-37, 19 May 2015, Florence (Italy). (査読有)
DOI: 10.1109/SPRO.2015.14

⑥ 永井晃人，神崎雄一郎，門田暁人，保護コードの自然さに着目した命令カムフラージュ，情報処理学会第 77 回全国大会講演論文集 (講演番号 1L-03)，2015 年 3 月 17 日，京都大学 (京都市)．

⑦ 松田篤和，神崎雄一郎，門田暁人，コード断片の不自然さの比較による保護機構の発見困難さの評価，情報処理学会第 77 回全国大会講演論文集 (講演番号 1L-02)，2015 年 3 月 17 日，京都大学 (京都市)．

6. 研究組織

(1) 研究代表者

神崎 雄一郎（KANZAKI, Yuichiro）

熊本高等専門学校・人間情報システム工学科・准教授

研究者番号：90435488

(2) 研究分担者

なし

(3) 連携研究者

なし