社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－DPS－122（6）
2005－CSEC－28（6）
2005／3／22

# デモジュレーションを用いた等価検証による自己改変型コードの検出

安藤類央　武藤佳恭
慶應義塾大学政策メディア研究科
〒２５２０８１６　神奈川県藤沢市遠藤５３２２
{ruo,takefuji}@sfc.keio.ac.jp

あらまし　冗長なアセンブラや分岐命令の挿入によりパターンマッチを回避する自己改変型のコードを、デモジュレーションを用いた形式的検証により検出する方法を提案する。提案手法では、検出対象となるコードを支持集合に加え、等価代入を繰り返すことで冗長性が付加される前のコードとの間に節矛盾を生じさせることで検出を行う。デモジュレータには、検出対象のアセンブラに固有の性質を、ヒューリスティックに記述することができる。これにより、不正なアセンブラ固有の知識を反映させた効率的な検出方法の可能性を示す。

## Formal method using demodulation against metamorphic viral coding

Ruo Ando Yoshiyasu Takefuji
Graduate School of Media and Governance,Keio University,
5322 Endo Fujisawa, Kanagawa, 252 Japan
{ruo,takefuji}@sfc.keio.ac.jp

In this paper we present a equivalence checking method applying demodulation for detecting　metamorphic viral coding. For effective searching, SoS (set of support) strategy is applied. Target code is added to the list of set of support with repeating equality substitution to generate　unit conflict. Demodulators could be described from the experience specified for skill developer.　This system enabled us to detect metamorphic code effectively in the point that we control scanning with demodulators heuristically adopted.

## １．はじめに

今世紀の初頭から、電子政府、電子商取引が本格的に実現化しつつあることから、インターネットは社会インフラの基幹を支える技術になりつつある。情報通信産業内に限らず、企業は情報化を進めることでコストの削減を図ることが通例となってきている。オンラインチケット、電子チェックイン業務、そしてATMなどの、ユーザが直接関与する機能を停止させた２００３年１月Win32 SQLワームは、セキュリティ侵害が顧客や利用者に及んだという点で、不正コードが社会問題化した最初のケースであると指摘されている。ウィルスが社会問題化した最初の事件と言える。また、多形態型（ポリモーフィック）や自己改変型（メタモーフィック）といった、パターンマッチを回避する手法が、常用されるようになってきており、これらの回避技術に対応する効果的な検出方法が求められている。

## ２．関連研究

不正プロセスの検出には、静的に解析をする方法と、動的に検出する方法があり、前者は、対象となるコードがメモリにロードされる前に、データマイニングやヒューリスティックなどの手法を使って解析する方法で、後者はコードをメモリにロードし、シミュレーションあるいはランタイムで検出を行う手法である。静的検出法は、プログラムを実行せずに検出が可能な方法であり、パターンマッチング法を中心として、比較方式とヒューリスティック方式に分かれる。コンペア法／チェックサム法／インテグリティチェック法[4]は、未感染のプログラムの情報を格納しておき、監視対象のコードの変化を検出する。パターンマッチング法[5]は、感染コードの特徴的なバイトコードが、対象となるプログラムに存在するか検査する方式である。ヒューリスティック方式[6]は、感染後の振る舞いを引き起こすと見なされ易いコードを解析することで、プログラムの異常を検出するものである。一般に、静的検出法は、ステルス技法の施されたコードや、ポリモーフィック／メタモーフィック化したコードに対しては、解析検出が困難であるという欠点が指摘されることが多い。ビヘイビア法は、ダイナミックヒューリスティック法とも呼

ばれ、監視対象となるコードを実際に動作させ、あらかじめ定めておいたルールに適合する挙動を検出する方法であり、実行環境は実マシンあるいはエミュレータ両方の場合がある[7]。

## ３.検出対象の定義

メタモーフィックコーディング[2]とは、冗長なアセンブラの挿入や置換により、同じ機能を持つが別の命令とレジスタの組み合わせから構成されるコードを再コンパイル／リンクすることで、シグニチャのパターンマッチを回避する手法である。メタモーフィックコーディングには、大きくわけて、１）レジスタを置換するタイプ、２）マジックワードを分解、変換するタイプ、３）ジャンプ命令の挿入により、命令順序や、エントリポイントを変更するタイプの３つがある。以下に、それぞれのタイプについて詳述する。

１）レジスタ置換型

リスト１は、レジスタを置き換えることで、操作後の各レジスタの状態は同じになるが、経由するレジスタが異なるため、シグニチャが変換するケースを示した。

```
POP EDX
MOV EDI, 0008H
MOV ESI,EBP
MOV EAX 000DH
ADD EDX, 005FH
MOV EDX,[EDX]
MOV [ESI+EAX*0000CCC9,EBX]

POP EAX
MOV EDX,0008H
MOV EDX,EBP
MOV EDI,000DH
ADD EAX,005FH
MOV ESI,[EAX]
MOV [EDX+EDI*0000CCC9],ESI
```

**リスト１：レジスタ置換型**

２）マジックナンバーの書き換え型

リスト２では、ＷＩＮＤＯＷＳオペレーティングシステム内での特定のＡＰＩのアドレスや、攻撃が成功した際に特定のアプリケーションを実行する機械語をシグニチャを検出する方法に対して、このようなマジックナンバーを変更して、検出を回避する方法を示した。

```
MOV DWORD PTR [ESI] ,11000000H
MOVDWORD　PTR
[ESI+0004],110000FFH
MOV EDI,11000000H
MOV [ESI],EDI
POP EDI
PUSH EDX
MOV DH,40
MOV EDX,110000FFH
PUSH EBX
MOV EDX,EBX
MOV [ESI+0004],EDX

MOV EDX,11000000H
MOV [ESI],EBX
POP EDX
PUSH ECX
MOV ECX,11000000H
ADD ECX,000000FFH
MOV [ESI+0004],ECX
```

**リスト２：マジックナンバー書き換え型**

３）制御順序の変更型

リスト３は、ジャンプ命令を利用することで、各オペレーションコードの順序を変えると語同時に、ジャンプ命令直後に実行されないコードを挿入し、パターンマッチを回避する方法を示した。

```
INSTRUCTION_A
INSTRUCTION_B
INSTRUCTION_C:

LABEL_2:
INSTRUCTION B
JMP
FAKE INSTRUCTIONS
LABEL_3:
INSTRUCTION_C
START::
LABEL_1:
INSTRUCTION_A
JMP
FAKE INSTRUCTIONS
```

**リスト３：制御順序の変更型**

以上、メタモーフィックウィルスが示す代表的な手法を挙げたが、現状ではこれらのテクニックに対応する決定的なヒューリスティックスキャンは存在せず、ウィルスソフトウェアベンダーの公表では、検出率は概して７０％から８０％前後と言われている。

## ４．提案手法

### ４．１　デモジュレーションと等価検証

等価検証とは、設計済みの回路やソフトウェアなどが意図した仕様をみたしているかチ

ェックするもので、シミュレーションによる検証と形式的検証の２種類がある。前者は、すべての入力パターンに対して望み通りの出力が得られるか確かめる方法である。形式的検証は、その設計の調べたい性質を定式化し、その性質を設計が満たしているか確かめるものである。本論文でのデモジュレーションを用いた検証では、形式的検証に分類される。

*T ← EQUAL(R,S)*

デモジュレーションは等価代入の一種であり、ここで、*EQUAL(R,S)*をデモジュレータと呼ぶ。このデモジュレータを項 T に適用するとき、T が R か S いずれかの具体例であることが条件となる。デモジュレーションによる操作は等価代入により対象となる情報を簡略化または正準化することが目的である。等価代入を起こった後の節は、デモジュラントと呼ばれる。等価代入を行う検証の特徴は、対象とする情報の性質を反映したデモジュレータを追加することにより、より効率的な情報の検索が行えるところにある。本論文では、この手法を用いて自己改変された２つのアセンブラのコードについて、等価検証を行う。

## ４．２　アセンブラの定式化

デモジュレーションによる検出を行うためには、検出対象となるコードについて、メタモーフィック化した後と以前のアセンブラのコードを、等価検証が可能なように、定式化する必要がある。本論文では、自己改変する前のコードを、

```
Loop:
        POP ECX
        JECXZ   Illegal_Code
        MOV ESI,ECX
        MOV EAX,0D601H
        POP EDX
        POP ECX
        CALL EDI
        JMP Loop
```

**リスト４：自己改変前のコード**

とした。検証を行うためのプログラムの表現方法については、出力を入力の関数として表す方法と、プログラムの構造自体を記述する方法がある。ここでは、後者の方法を採り、前節で述べた命令順序の変更による改変方法に対応するために、以下のような定式化を行った。

Asm(I3,MOV(ESI,ECX))

ここで、Asm はデモジュレーションを行うための関数であり、I3 は命令の順序を表す。検証方法のフレームとしては、以上の定式化したコードについて、支持集合戦略を基にしたデモジュレーションを行う。

## 4. .3 単位矛盾と検出

上で述べたデモジュレーションが、終了するためには、検索の過程で、単位矛盾を生じさせる必要がある。単位矛盾とは、厳密には改変前と改変後のアセンブラコードを形式化した節同士が共通するリテラルを持ち、双方のリテラルが逆の符号を持ち、単一化可能な状態をさす。本論文では、メタモーフィック化される前のコードを否定した負の節を設定し、デモジュレーションによって同じリテラルを持つ節が生成された時点で等価性が証明されたとし、検出を終了する。

今回、定理証明系には、アルゴンヌ国立研究所で開発された、Otter(Organized Techniques for Theorem-proving and Effective Research)を用いた。同ソフトウェアは、一階述語論理の定理証明系には安定した性能を示していることで知られており、ベンチマークとして比較評価されることが多い。リスト５では、提案手法を、OTTER で使われる形式で記述したものである。

```
１）支持集合節：
｛改変後のコード｝

２）通常の節集合：
｛改変前のコード｝

３）デモジュレータ：
｛形式化したアセンブラコードに、等価代入を行うための式｝
```

**リスト５：単位矛盾を用いた検出**

## 5. 評価実験

評価実験では、不正なコードのシグニチャとなるアセンブラのコードの前後にデバッグ命令を挿入し、例外処理としてデモジュレーションによる等価検証を行うことで、自己改変型のコードの検出を行った。

## ５．１　プロセス構造ルーチン

自己改変しているコードの前後で、デモジュレーションを行うために、本論文ではプロセス構造ルーチンを用いてロード時に登録されるコールバック関数を通じて上の操作を行う。ここで、プロセス構造ルーチンとは、実行ファイルがメモリにロードされプロセスとなる際に、コールバック関数やブレークポイントを挿入するために用いるドライバ関数のことを指す。

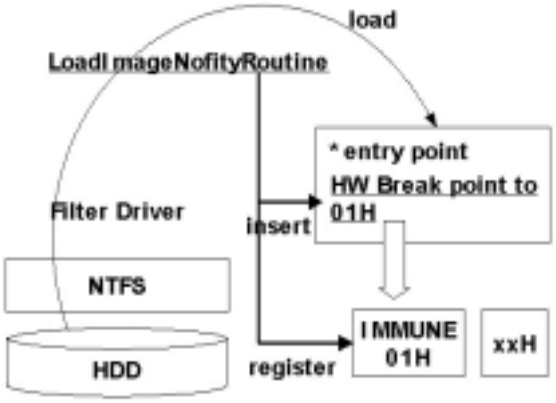


図１　プロセス構造ルーチンを使ったＩＮＴ１命令の挿入

図１は、コールバック関数内での操作を示したものである。提案システムでは、プロセス構造ルーチンを用いて、実行ファイルのロード時にコールバック関数を登録する。Win32のドライバ関数を用いると、イメージが実行時にロードされるときに呼び出されるコールバック関数を登録することができる。

```
Vvoid LoadImageNotifyRoutine{
        PUNICODE_STRING FullImageName,
        HANDLE ProcessId,
        PIMAGE_INFO ImageInfo
}
```

上の関数は、提案システムで LoadImageNotifyRoutineを示したものである。同コールバック関数内で行われる作業は、エントリポイントのアドレスの取得し、ハードウェアブレークポイントを挿入するというものである。これにより、自己改変したコードの前後で、デモジュレーションを行うための例外ハンドラが呼び出されることになる。この改良例外ハンドラ内で、次節で説明するデモジュレーションによる等価検証が行われる。

### ５．２　デモジュレーションによる検出

評価実験では、リストを以下のようにメタモーフィック化したコードをサンプルとして、等価検証を行った。

```
Loop:
        POP ECX
        NOP
        JMP ROUTINE1
ROUTINE3:
        CALL EDI
        XOR EBX,EBX
        BEQZ N2
N2:
        JMP Loop
        JMP L4
ROUTINE2:
        NOP
        MOV EAX,0D601H
        POP EDX
        POP ECX
        NOP
        JMP ROUTINE3
ROUTINE1:
        JECXZ Illegal_Code
        XOR   EBX,EBX
        BEQZ N1
N1:
        MOV ESI ECX
        JMP ROUTINE2
ROUTINE4:
```

リスト６：自己改変後のコード例

ここで、前節で述べた定式化のための表現Asm(x)を用い、各ルーチンを以下のように表現する。

```
ROUTINE1:
        I4(I3,JECXZ(SFModMark))
        I5(I4,XOR(EBX,EBX))
        I6(I5,BEQZ(N1))
```

リスト７：アセンブラの形式化

リストは、命令 I1 から I19 までが辞書的順序付けにしたがってデモジュレーションによって書き換えられる。これにより、JMP 命令によるオーダーが、変形前の状態に戻される。次に、冗長な命令と、JMP 命令を消去するために、次のようなデモジュレータを追加する。

```
asm(x,JMP(y))=asm(x).
asm(x,BEQZ(y))=asm(x).
asm(x,XOR(EBX,EBX))=asm(x).
asm(x,NOP)=asm(x).
asm(NOP,x)=asm(x).
```

リスト８：デモジュレータ

評価実験では、３１回の等価代入の後、前節で扱ったマッチングするアセンブラとの節矛盾がおき、リストは、リストのメタモーフィック化したコードであることの検証に成功した。

## 6. まとめと今後の展望

本論文では、デモジュレーションを用いた等価検証を、冗長なアセンブラとルーチンの挿入により、パターンマッチを回避する自己改変型コードの検出へ適用した。評価実験では、冗長

性が付加される前の、アセンブラコードを支持集合に加え、検出対象となるコードに等価代入を繰り返すことで、単位矛盾を生じさせ、提案手法の検出時の有効性を実証した。

今回、自己改変する前のアセンブラの構造を記述したが、今後の課題としては、レジスタの値に注目し、各レジスタの出力を入力の関数として表現し、等価検証を行うことが考えられる。これにより、３節で述べたマジックナンバー改竄型の手法にも対応できると想定される。また、自己改変型コードはエントリポイントの偽装に特徴があり、処理対象とするコードをどのようにして区切っていくかについても課題が残っている。またデモジュレーションにはたくさんの計算戦略が提案されており、等号調整代入（パラモジュレーション）も含め、その他の戦略の適用を工夫する余地が残っている。

**謝辞**

本研究は文部科学省２１世紀ＣＯＥプログラム次世代メディア・知的社会基盤の拠点形成活動の一環である。また、助言・協力をいただいたサイエンスパーク株式会社の野崎隆氏、三浦秀明氏に謝意を示す。

参考文献

[1]独立行政法人:情報処理推進機構:未知ウィルス検出技術に関する調査,
１５情経第１６７５号(2004)
[2] Szor, Peter and Ferrie, Peter. "Hunting for Metamorphic." Virus Bulletin Conference, September 2001.
[3] Stephen Pearce, “Viral Polymorphism”, paper submitted for GSEC version 1.4b,2003.
[4]Gene H. Kim and Eugene H. Spafford:Tripwire A File System Integrity Checker,
ACM Conference on Computer and Communications Security , pp. 18-29(1994).
[5]Roesch, M:Snort - lightweight intrusion detection for networks.
Proceedings of Thirteenth Systems Administration Conference (LISA '99), pp. 229-238(1999).
[6]Symantec Corporation: Bloodhound Technology.
http://securityresponse.symantec.com/
[7]三宅崇之,白石義明,森井昌克,”仮想ネットワークを使った未知ウィルス検知システム,”情報処理学会研究報告 ,No.022-016(2003).
[8]Larry Wos, George A. Robinson, Daniel F. Carson, Leon Shalla: The Concept of Demodulation in Theorem Proving. J. ACM 14(4): 698-709 (1967)
[9]Larry Wos: The Problem of Demodulation During Inference Rule Application. J. Autom. Reasoning 9(1): 141-143 (1992)
[10]M.Christodorescu,Somesh Jha,”Testing malware detectotos”, proceedings of International symposium on software testing and analysis,Boston,MA,2004.
[11]Akira Mori,”Detecting Unknown Computer Viruses - A New Approach”, ISSS 2003: 226-241