**SCIS 2007** The 2007 Symposium on Cryptography and Information Security
Sasebo, Japan, Jan. 23-26, 2007
The Institute of Electronics, Information and Communication Engineers

# Equational reasoning を用いた obfuscated decipher routine のパラメータ検出

安藤類央*

**あらまし** ソフトウェアの難読化と暗号化は、耐タンパーソフトウェア技術または不正コードの検出回避技術双方において重要になってきている。本論文では難読化された復号ルーチンの解析に、Equational reasoning を適用し、各種パラメータを検出するための方法論を示す。提案手法では、難読化された実行コードを逆アセンブルし、節表現の形式に変換する。変換された節群に対して、レゾリューションにより縮退（de-obfucsation) を行う。次に、検出されたプログラム構造を元に等価代入を行い、パラメータを検出する。等価代入の操作にはデモジュレーションとパラモジュレーションの２つを用いた。また、この過程で生成された節数を元に、縮退やパラメータ検出の難易度の指標を作成することが可能である。評価実験では数体の obfuscated decipher routine を持つ検体を、提案手法を用いて検出した。

**キーワード** 難読化と暗号化、obfuscated decipher routine、equational reasoning、アセンブラコード、FoL 処理系

## 1 はじめに

ソフトウェアの暗号化と難読化は、従来耐タンパー性を確保するための技術として注目されてきたが、近年不正コードの検出を回避する方法として適用されることが多くなり、重要性を増してきている。ソフトウェア難読化の技術は、比較的新しいという面もあり、難読化のコードの自動生成の研究、あるいは特殊なノウハウが所謂アンダーグランドと言われるインターネットのサイトで共有されることはあるが、それらを統合的・科学的に分析し、評価するための研究は行われることは現時点では稀である。本論文では、難読化された復号ルーチンに対して、一階述語論理の定理証明系を用いて、コードの縮退とパラメータ（鍵、アドレス、カウンタ等）の検出を行うための方法論を示す。

## 2 関連研究

コンピュータウィルス（ワーム、トロイを含む）とその検出の理論的な定義と議論は [1][2] で行われている。Symantec Corporation が２００１年に W32.Simile についての解説を出した頃から、metamorphic, polymorphic といった暗号化と難読化を行う不正コードの研究が活発に行われるようになった [5][6][7]。[8] ではモデル検査の適用が、[9] では attack graph を用いた手法が検討されるようになっている。Reodering instructions についての対応は [10] で議論されている。

* 独立行政法人情報通信研究機構 情報通信セキュリティ研究センター, 東京都小金井市貫井北町 4-2-1, 4-2-1 Nukui-Kitamachi, Koganei, Tokyo 184-8795 Japan, ruo@nict.go.jp

## 3 提案手法

提案手法は、[1] 難読化された復号ルーチンのコードの縮退、[2] 鍵、各種アドレス、カウンタのパラメータの特定という２段階からなる。[1] にはレゾリューション、[2] には等価代入 (equality substitution) という手法を適用する。

### 3.1 コードの縮退

ここでのコードの縮退とは、機能的に等価であるが複雑度（冗長度）の高いプログラムを、より簡潔なプログラムに変換することを指す。de-obfuscation や simplification にあたる操作である。図は、提案手法の１段階目を示したものである。例えば、図の右段にある操作は、

```
-(1-1) | -(1-3) | 1.
if (1-1) and (1-3) is true, then 1.
```

は、遷移公理 (Transition axioms）と呼ばれ、難読化されたコードを簡素化するのに用いる。例えば、

```
-(1-1) | -(1-3) | 1. :
if (1-1) and (1-3) is true, then 1.
-(mov dword_1 0h) | -(mov edx dword_1)
| mov edx 0h.
```

は、register substitution という、冗長なレジスタ代入により難読化を行うテクニックに対応するものである。

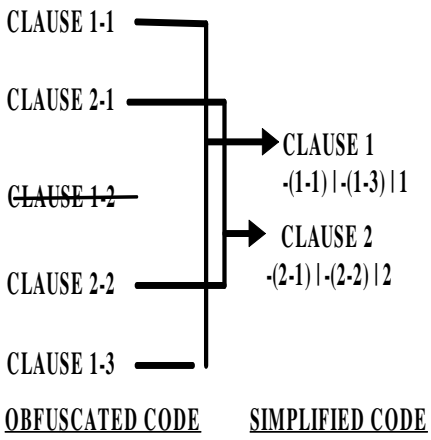


図 1:

### 3.2 パラメータの検出

本論文では、パラメータの検出に equational reasoning という手法を適用する。一階述語論理の定理証明系では、等価代入 (equality substitution) がこれに含まれる。具体的には、demodulation と paramodulation という操作によって等価代入を行う。

```
fact : f(g(x),x).
fact : equal(g(a),b).
conclusion f(b,a).

fact : equal(data_16e,514Bh).
fact : mov(reg(ah),const(data_16e),63,time(1)).
conclusion :
mov(reg(ah),const(514Bh),63,time(1)).
```

上の２つのコードは、デモジュレーションとその適用例を表したものである。この操作は、デモジュレータ（equal 節）を、fact 節に適用し、等価代入を行い、対象となる節の簡素化を行う。

```
fact: mov(reg(ah),const(2Ch),162,time(1)).
fact: mov(reg(bx),reg(ah),300,time(1)).
fact: decrypt(reg(dx),reg(bx),431,time(1)).
/* decrypter */

-mov(reg(x),const(y),z,time(1)) | x=const(y,z).
conclusion : decryptor(reg(dx),
key(const(2Ch,162),431,time(1)).
```

上の２つのコードは、パラモジュレーションとその適用例を示したものである。デモジュレーションと比較してより多様な節形式に利用することができる。デモジュレーションが effectivness を主眼をしているのに対し、この操作では概して、完全性（completeness) が保たれるとされる。デモジュレーションが一回の等価代入を行って停止するのに対し、パラモジュレーションは使われなかった節はＯＲを取って保存されるため、等価代入によって推論プロセスを進めていくことが可能である。

### 3.3 節表現への変換

図は、難読化された実行ファイルから、定理証明系が処理できる節形式の表現 (clausal representation) に変換し、検出を行うまでのフローチャートである。実行ファイルを SOUCER や IDA Pro などのソフトウェアで逆アセンブルし、以下のような形式に変換する。

```
instruction(operand1(x),operand2(y),z,time(1)).
```

ここで、instruction は opcode（命令)、operand1、２は命令の引数、z はアドレス、time() は、推論の過程によって処理された（実行された）回数である。

## 4 適用推論技法

### 4.1 支持集合戦略

支持集合戦略 [7] は、１９６５年に Wos らによって提案されたものである。この計算戦略は制限戦略の１つで、自動推論プログラムに目標とする解空間に関係ないところを探索せずに、対象としている問題に集中させるようにする。節集合 S、T があり、S-T が充足可能であるとき、T は S の支持集合である。このとき、支持集合に属さない節同士では導出を行わず、支持集合に属する節との間で、導出を行う方針を支持集合戦略という。

### 4.2 超導出

超導出 [8] は１９６５年に Robinson らによって提唱された手法で、通常の導出系の手法では１対の節から順次導出を行うのに対して、２個以上の節に対して導出を行う。超導出の意味は、何段階もの２項導出にあたる作業を１つにまとめたもので、通常の２項導出に比べて、多くの導出が起こるという事を指す。

### 4.3 包摂

定理証明を用いた推論プロセスでは、目標とする節を導出する過程で、いくつかの節が保持され、新しい節が生成された時点で、過去に保持された節との間で、改めて定理が適用される。この保持されている節のうち、より一般的な節を残す処理を包摂 [10] という。

### 4.4 デモジュレーション

デモジュレーション [9] とは、あらかじめ等価代入を行うための節を定理証明系に加えて、処理節群の簡略化あるいは正準化を行う処理である。デモジュレーションは、それ自体で冗長な命令、あるいは、MOV 命令によるメモリ、レジスタ、変数間の転送によるプログラムの状態遷移を簡略化するのに有効である。

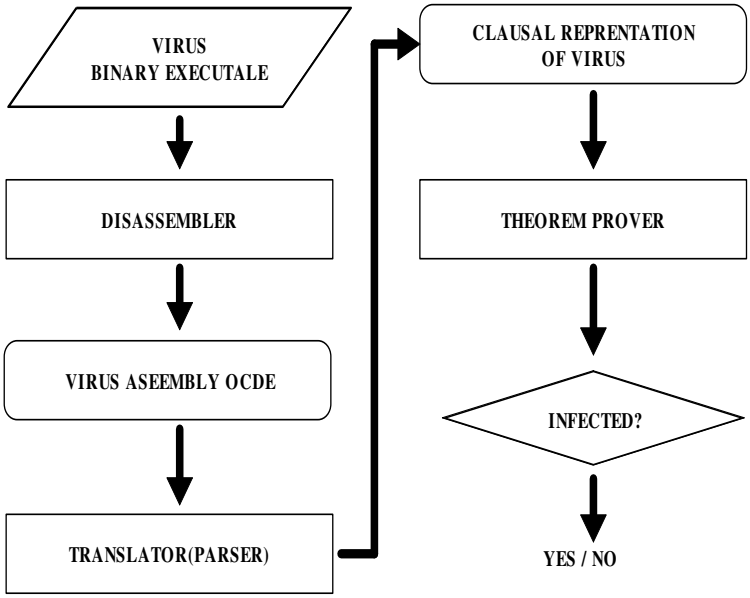


図 2: 実行ファイルから節形式への変換と検出

### 4.5 パラモジュレーション（等号調整代入）

パラモジュレーション（等号調整代入）は、デモジュレーションと同じく等価代入の操作である。デモジュレーションが effeciency を重視して設計されているのに対し、パラモジュレーションは概して推論過程で完全性が保持される。パラモジュレーションはデモジュレーションの一般形である。デモジュレーションは一回等価代入が完了すると停止するが、パラモジュレーションでは関連する節はＯＲを取って保持される。そのため、等価代入のみで状態遷移を起こして、推論プロセスを進めることが可能である。

## 5 評価実験

### 5.1 サンプルコード

今回、提案システムの評価実験用のコード生成には、SMEG (simulated metamorphic encryption generator)[4] を用いた。SMEG は、有名なメタモーフィックコード生成器の１つであり、SMEG.Pathogen や SMEG.Queeg などのウィルスは同プログラムをエンジンとして組み込んでいる。評価実験では、SMEG の５体の難読化された (obfuscated) ウィルスコードを生成させ、定理証明器によって復号ルーチンを検出する際に生成された節数などを比較した。SMEG によって生成されるウィルス群は以下の３種のタイプのコードに分類される。

```
loop_start
mov data [address]
decrypt data key
mov [address] data
inc address
dec counter
test counter counter
jmp loop_start
```

上のコード (type I) は、もっともベーシックなタイプで、生成されるコードの半分程の割合を占めるものである。ＭＯＶ命令で暗号化されたペイロードを転送し、復号したあとに格納元のアドレスに戻し、各種パラメータを更新する。

```
loop_start
decrypt [address] key
inc address
dec counter
test counter counter
jmp loop_start
```

上のコード (type II) は、データ転送と復号に間接アドレッシングを用いるもので、ＭＯＶなどの転送命令を使わず、直接アドレスを指定し、格納されているペイロードを復号する。

```
loop_start
xchg data [address]
decrypt data key
xchg [address] data
inc address
dec counter
test counter counter
jmp loop_start
```

上のコードは、type ＩでのＭＯＶ命令の代わりにＸＣＨＧ (exchange) 命令を用いたものである。

### 5.2 検出パラメータ

検出パラメータは、ペイロードのアドレス、鍵、復号ルーチンのスタートアドレス、そしてカウンタの４つである。

```
define A address_of_payload
define B key
define C address_loop_start
define D counter

address_loop_start
   payload_transfer(A)
   decryptor(B)
   parload_transfer(A)
   branch(D)
   goto_start(C)
```

上は、復号ルーチンの構造の一例を簡略したものである。ループに入る前に、define やＭＯＶ命令を使ってパラメータを規定する。

### 5.3 実験結果

前節までで述べた、復号ルーチンのパラメータ検出の際に生成された節数を表に示す。表は、それぞれ生成されたコード、そのコードのタイプ（１から３、前節参照)、そして生成された節数を示している。難読化されたコードのタイプや検出するパラメータによって生成された節数が予想以上に上下することが分かった。また、実際に解析を行った経験からは、同実験内に限り、難読化の程度を適切に示していると想定される。

### 5.4 まとめと今後の課題

ソフトウェアの暗号化と難読化は、従来耐タンパー性を確保するための技術として注目されてきたが、近年不正コードの検出を回避する方法として適用されることが多くなり、重要性を増してきている。ソフトウェア難読化の技術は、比較的新しいという面もあり、難読化のコードの自動生成の研究、あるいは特殊なノウハウが所謂アンダーグランドと言われるインターネットのサイトで共有されることはあるが、それらを統合的・科学的に分析し、評価するための研究は行われることは現時点では稀である。本論文では、難読化された復号ルーチンに対して、一階述語論理の定理証明系を用いて、コードの縮退とパラメータ（鍵、アドレス、カウンタ等）の検出を行うための方法論を示した。提案手法では、難読化された実行コードを逆アセンブルし、節表現の形式に変換する。変換された節群に対して、レゾリューションにより縮退 (de-obfucsation) を行う。次に、検出されたプログラム構造を元に等価代入を行い、パラメータを検出する。等価代入の操作にはデモジュレーションとパラモジュレーションの２つを用いた。また、この過程で生成された節数を元に、縮退やパラメータ検出の難易度の指標を作成することが可能である。評価実験では数体の obfuscated decipher routine を持つ検体を、提案手法を用いて検出し、定量的な評価を行うことを可能にした。今後の課題としては、より精緻かつ高速なパラメータ検出法や、バイナリコードの解析などが挙げられる。

## 参考文献


[1] Computer viruses: from theory to applications. IRIS International series, Springer Verlag, ISBN 2-287-23939-1, juin 2005. English edition of the book on computer viruses.

[2] Diomidis Spinellis. :Reliable identification of bounded-length viruses is NP-complete. IEEE Transactions on Information Theory,January 2000 :280-284.

[3] Peter Szor and Peter Ferrie.:Hunting for Metamorphic. Virus Bulletin Conference, September 2001: 123-144.

[4] Stephen Pearce, ”Viral Polymorphism”, paper submitted for GSEC version 1.4b,2003.

[5] ”Network-level polymorphic shellcode detection using emulation” Michalis Polychronakis, Kostas G. Anagnostakis and Evangelos P. Markatos DIMVA 2006

[6] Semantics-Aware Malware Detection Mihai Christodorescu, Somesh Jha, Sanjit A. Seshia, Dawn Song, Randal E. Bryant IEEE Security and Privacy 2005

[7] Static Analysis of Executables to Detect Malicious Patterns (2003) Mihai Christodorescu and Somesh Jha 12th USENIX Security Symposium, August 2003

[8] Hao Chen, Drew Dean, and David Wagner.Model checking one million lines of C code. In Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS), pages 171–185, San Diego, CA, February 2004.

[9] O.Sheyner, J.Haines, S.Jha, R.Lippmann, and J. M. Wing, ”Automated Generation and Analysis of Attack Graphs”, IEEE Symposium on Security and Privacy , April 2002.

| generated code #1 | | | | |
|---|---|---|---|---|
| Type I | Branch | Decrypt | Loop | Transfer |
| clauses generated | 3378 | 30480 | 4292 | 30471 |
| para_from generated | 1358 | 15935 | 1799 | 15935 |
| para_into generated | 1463 | 13366 | 1826 | 13362 |
| generated code #2 | | | | |
| Type II | Branch | Decrypt | Loop | Transfer |
| clauses generated | 1158 | 1466 | 1258 | 719 |
| para_from generated | 423 | 435 | 435 | 322 |
| para_into generated | 390 | 495 | 431 | 158 |
| generated code #3 | | | | |
| Type III | Branch | Decrypt | Loop | Transfer |
| clauses generated | 2751 | 10184 | 3072 | 909 |
| para_from generated | 1186 | 5330 | 1436 | 335 |
| para_into generated | 803 | 3932 | 1008 | 185 |
| generated code #4 | | | | |
| Type I | Branch | Decrypt | Loop | Transfer |
| clauses generated | 808 | 2890 | 923 | 703 |
| para_from generated | 255 | 1125 | 268 | 255 |
| para_into generated | 271 | 1170 | 337 | 212 |
| generated code #5 | | | | |
| Type I | Branch | Decrypt | Loop | Transfer |
| clauses generated | 6327 | 11990 | 9903 | 3235 |
| para_from generated | 2669 | 3532 | 2748 | 1049 |
| para_into generated | 2227 | 3474 | 2686 | 892 |

表 1: 等価代入 (equational reasoning) によるパラメータ検出時に生成された節数

[10] Arun Lakhotia, Moinuddin Mohammed, "Imposing Order on Program Statements to Assist Anti-Virus Scanners", In Proceedings of Eleventh Working Conference on Reverse Engineering, Delft, The Netherlands, November 2004, pp. 161-170.

[11] Matias Madou, Bertrand Anckaert, Patrick Moseley, Saumya Debray, Bjorn De Sutter, Koen De Bosschere. Software Protection through Dynamic Code Mutation. Proc. Int. Workshop on Information Security Applications, Aug. 2005.

[12] Arun Lakhotia, Moinuddin Mohammed: Imposing Order on Program Statements to Assist Anti-Virus Scanners. WCRE 2004: 161-170

[13] Simulated Metamorphic Encryption Generator http://vx.netlux.org/vx.php?id=es06

[14] Larry Wos, George A. Robinson, Daniel F. Carson, Leon Shalla: The Concept of Demodulation in Theorem Proving. J. ACM 14(4),1967:698-709

[15] Larry Wos: The Problem of Explaining the Disparate Performance of Hyperresolution and Paramodulation. J. Autom. Reasoning 4(2): 215-217 (1988)

[16] Larry Wos, George A. Robinson, Daniel F. Carson: Efficiency and Completeness of the Set of Support Strategy in Theorem Proving. J. ACM 12(4), 1965:536-541

[17] IDA Pro disassembler: http://www.datarescue.com/

[18] OTTER automated deduction system, http://www.mcs.anl.gov/AR/otter/

[19] Intel Corporation: IA-32 IntelR Architecture Software Developer's Manual, Volume 2B: Insruction Set Reference N-Z,2004.