仮想マシンモニタを用いたOSコードの秘匿化

佐久間 充† 大山恵弘†

1. はじめに

プログラムの処理内容を隠したい場合にはバイナリコードのみを公開するが、リバースエンジニアリング[1)]により、バイナリコードからプログラムの処理内容を解析することは可能である。リバースエンジニアリングには、失われた情報を復元する利点としての目的もあるが、例えば知的財産を侵害する悪意を持った利用としての目的も存在する。

そこで、プログラムの難読化[2)3)]による防衛手法が提案されている。しかしそれらの手法は、あくまでプログラム解析の手間を増加させるようにコードを変換するのであって、努力次第ではやはり難読化したプログラムの処理内容も解析されてしまう。

コードの処理内容の解析を防ぐためには、コード全体をユーザに与えない手法、つまりコードを秘匿化する手法がより効果的である。

そこで本研究では、OSコードの秘匿化を可能にするシステムの構築を目的とする。本システムでは、一部の命令が秘匿化されたOSコードを、特殊な仮想マシンモニタ(VMM)上で動作させる。本システムは、通常のOSコードを受け取り秘匿化したOSコードを返す「変換機構」と、その秘匿化したOSを動作させる「拡張VMM」から構成される。

VMMとは一般に、仮想的な計算機環境(VM)の生成や操作を行うプログラムのことである。VMMの特徴として、VM上でOSが動作可能であることと、VM上で実行される命令のうち特定の命令(hlt, cpuidなど)をインターセプトし、エミュレーション実行可能であることを本研究では利用する。本研究では、通常のVMMを、秘匿化したOSを動作させる機能で拡張した。以降ではそれを拡張VMMと呼ぶ。

2. システムの設計

本システムの設計を説明する。

2.1 変換機構

変換機構は、秘匿化していないOSコードを受け取り「秘匿化したOSコード」と「対応表」を作成するシステム(図1)である。

秘匿化したOSコードとは、公開したくない命令(実命令)をhlt命令(秘匿命令)で上書きしたOSコード(図2)であり、対応表とは、実命令と秘匿命令の対応を記録した表(図3)である。対応表の具体的な内容は、秘匿命令で上書きされた実命令とそのメモリアドレスとなる。

2.2 拡張VMM

拡張VMMは、変換機構が作成した秘匿化したOSコードをVM上で動作させるためのVMM(図4)で

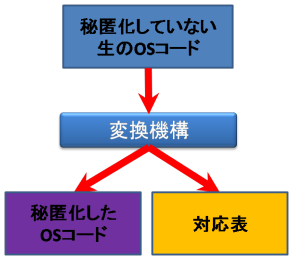

図1 変換機構

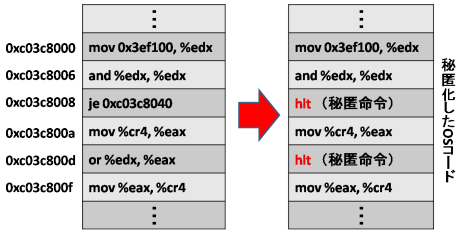

図2 OSコードの秘匿化

上書きされた命令（実命令）	上書きされたメモリアドレス
⋮	⋮
je 0xc03c8040	0xc03c8008
or %edx, %eax	0xc03c800d
⋮	⋮

図3 対応表

† 電気通信大学 電気通信学部 情報工学科

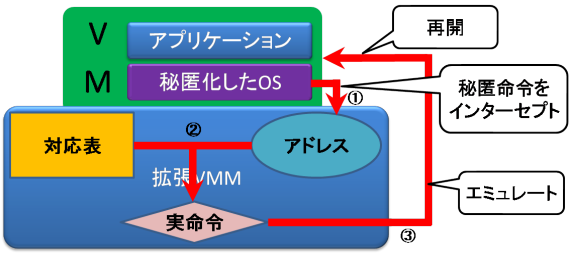

図 4 拡張 VMM による処理の流れ

ある。

拡張 VMM において、どの様な処理によって秘匿化した OS を動作可能にするかを説明する。VM 上で秘匿命令、つまり hlt 命令が実行された場合、拡張 VMM はその命令をインターセプトする。拡張 VMM は続いて、対応表と、インターセプトした秘匿命令が存在するメモリアドレスから、その秘匿命令に対応する実命令を調べる。そしてその実命令をエミュレート (または CPU で直接実行) した後に、OS を再開させる。これらの処理によって、秘匿化した OS は、秘匿化された状態のまま VM 上で動作可能となる。

3. 現状と今後の課題

システムの設計を行った。現在は VMM の実装を行っている。今後は、VMM の実装と秘匿化処理の実装を完了させ、オーバーヘッドの計測などを通じてシステムの評価を行いたい。

参 考 文 献

1) Elliot J. Chikofsky, James H. Cross II: Reverse Engineering and Design Recovery. In *IEEE Software*, Vol.7, No.1, pp. 13-17, January 1990.
2) 門田 暁人, 高田 義広, 鳥居 宏次: ループを含むプログラムを難読化する方法の提案. 電子情報通信学会論文誌 D Vol.J80-D1 No.7, pp.644-652, 1997 年 7 月.
3) Igor V. Popov, Saumya K. Debray, Gregory R. Andrews: Binary Obfuscation Using Signals. In *Proceedings of 16th USENIX Security Symposium*, pp. 275-290, Boston, August 2007.