命令セットに含まれる自由度の評価とその応用

指導教員：市川周一　　学籍番号：043714 澤田豊志

1　はじめに

ウイルスやワームによる攻撃は大きな社会問題になっている．これらの不正プログラムは，多くの場合，バッファオーバーフロー等の脆弱性を利用して悪意ある命令列を実行させ，実行制御の乗っ取りを行う．

不正コードのメモリへの注入 (injection) を防止する手法として，実行時にメモリのシグネチャを検証する方法，メモリ全体を暗号化する方法，等が提案されている．しかしメモリアクセスにオーバヘッドが発生するため，顕著な性能低下が発生することが問題となっている．

一方，注入されたコードを無効化する手法としては，命令セットのランダム化が提案されている．ウイルス等の不正コードは，一般に特定のアーキテクチャ(例えば x86) を前提としているので，異なるアーキテクチャの計算機に感染することができない．Kc ら [1] は，(1) 命令ワードをビット転置すること，(2) 命令列に秘密の鍵を XOR すること，等で命令列 (バイナリ表現) に多様性を持たせることを提案した．Barrantes ら [2] は，実行イメージをメモリにロードする際に擬似ランダム数列と XOR することを提案した．これらはいずれも命令列を符号化するもので，厳密には命令セットのランダム化ではない．

本研究では，命令コードの符号化の自由度を利用して命令セットをランダム化する手法を提案する．本手法は応用を問わず適用可能だが，特に FPGA 等を用いた組込みシステムには適していると考えられる．

2　命令セットの自由度

命令セットアーキテクチャ(ISA) では，使用可能な命令のセットと，レジスタ等の使用可能な資源が定義されている．各命令には処理内容とオペランドが定義されており，それらのメモリ表現は命令形式によって定められている．各命令形式にはフィールドが定義されており，命令は opcode によって区別される．

同じ命令形式の 2 つの命令 A, B があったとき，その opcode の値を入れ替えた ISA を考える (図 1 上)．この 2 つの ISA は全く同じ命令形式と機能を持ち，実用上は同じ ISA であるといってよいが，命令 A と B のメモリ表現だけが異なっている．本研究では，このような符号化だけの相違を**パーソナリティ**と呼ぶことにする．ISA によっては，命令種別を opcode 以外の拡張フィールドで指定する場合もある (図 1 下の function)．その場合は，拡張フィールドの入れ替えによっても新たなパーソナリティが生成できる．

パーソナリティの異なるプロセッサはバイナリ非互換なので，ウイルス等の不正コードに一定の耐性を持つことになる．また，元になる ISA を知っていてもパーソナリティ(opcode 割当) を知らないと命令列バイナリを解読できないため，コードの偽造や解析が困難になる．一方，パーソナリティを知っていれば，バイナリ相互の変換は非常に容易である．パーソナリティ非依存のツール (コンパイラなど) が全て共有できるので，開発環境や OS を新規開発する必要はない．また，本手法は既存の手法 (メモリ暗号化，シグネチャ検証，Kc ら [1] や Barrantes ら [2] の手法) と共存可能である．

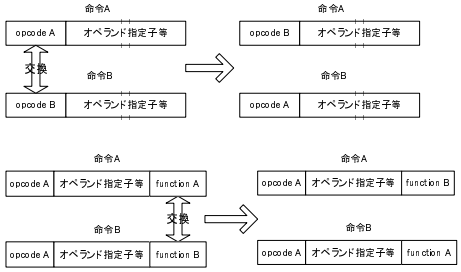

図 1: パーソナリティの生成

3　自由度の評価

各 ISA に対して生成可能なパーソナリティの数に応じて，攻撃への耐性が高まると考えられる．本研究では，特に組込み分野への応用を考慮して，MIPS32 (rel.1)，SH-3, Java VM, 8080 の 4 種の ISA に関して命令セットの自由度を評価した．MIPS32 は 32 bit 固定長命令，SH-3 は 16 bit 固定長命令，Java VM と 8080 は命令コード 8 bit の可変長命令である．評価にあたっては，ハードウェア化の容易さを考えて，同じ命令形式内の自由度だけを考慮し，未定義命令は自由度から除外し，オペランドやオペランド指定子の自由度は除外した．

各 ISA の自由度と，自由度を情報量 [bit] に換算した結果を表 1 に示す．4 つの ISA で総命令数に大きな差はないが，自由度は大きく異なっている．命令が長ければ自由度が大きいとは限らず，命令形式の直交性などが自由度に大きな影響を及ぼしている．Java VM の自由度が大きいのは，命令形式が 1 つしか存在せず，8 bit の命令コードに 201 の命令が定義されているためである (201!　10^{377})．一番自由度の少ない SH-3 でも，実用上充分な自由度が存在することがわかる．

参考として，命令種別毎の自由度 (の一部) を表 2 に示す．各 ISA で命令種別の分類・定義が異なっているため，この結果は厳密なものではない．例えば SH-3 のマニュアルでは，PREF(prefetch) 命令はデータ転送命令とシステム制御命令の両区分に含まれるが，ここではデータ転送命令に数えた．

4　おわりに

パーソナリティの異なる命令列を実行するには，(1) エミュレータで解釈実行する，(2) Transmeta 社の Crusoe のように実行時にエミュレートする，(3) 各パーソナリティに特化したプロセッサを実現する，などの方法がある．パーソナリティをハードウェアで実装する場合も，論理の相違はデコーダ部分にしか発生しない．FPGA などを利用すれば実装は比較的容易で，コストや性能オーバヘッドは少ないと考えられる．

パーソナリティは一種の写像なので，デコード部で表を切り替えることにより，複数のパーソナリティをもつプロセッサを実現することもできる．同じ仕組みで，ユーザ毎，プロセス毎，OS に別個のパーソナリティを与えることも可能である．複数パーソナリティを実装するプロセッサを設計する際，各パーソナリティを指定するために，最低限でも表 1 に示した情報量を格納するメモリが必要である．実際には実装の都合で，最低必要量の 2 倍程度は必要ではないかと考えられる．例えば Java VM をマルチパーソナリティ化するためには，$256 \times 8 = 2048$bit 程度の RAM で写像する必要がある．

参考文献

[1] G.S. Kc, A.D. Keromytis, V. Prevelakis: “Countering Code-Injection Attacks with Instruction-Set Randomization,” Proc. CCS’03, pp. 272–280, ACM (2003).

[2] E.G. Barrantes et al.: “Randomized Instruction Set Emulation,” ACM Trans. Information and System Security, vol. 8, no.1, pp.3–40 (2005).

表 1: 命令セット全体の自由度

	総命令数	自由度	情報量 [bit]
MIPS	170	2.34e+166	553
SH-3	188	1.63e+90	300
Java VM	201	1.59e+377	1253
8080	111	2.34e+136	453

表 2: 命令種別ごとの自由度（一部）

	算術演算	データ転送	分岐	システム制御	特権
MIPS	4.68e+21	1.29e+39	4.05e+18	2.00e+00	8.64e+03
SH-3	6.53e+28	9.06e+25	1.15e+03	5.76e+03	2.40e+01
Java VM	2.65e+32	8.50e+101	2.59e+22	1.00e+00	–
8080	2.30e+10	1.55e+25	3.05e+29	4.03e+04	–