命令セットアーキテクチャの多様化に関する研究

指導教官：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社会問題となっている．これらの攻撃の多くは，バッファオーバーフロー等を利用して不正な命令列を実行させるものである．

不正な命令列の実行を防ぐ方法として，メモリの暗号化や命令の実行時に認証を行なうといった手法が提案されている．しかし，これらの方法はメモリアクセス遅延の増大による性能低下が大きい．

一方，プロセッサごとに異なる命令セットを使い多様性を持たせるという方法がある．個々のプロセッサごとに命令セットが変わっていれば，攻撃者は対象の命令セットに対する知識がなければ攻撃がしにくくなる．このような命令セットをランダム化する手法の提案として Kc ら [1] は，命令列に鍵を XOR する手法と命令コードのビットを転置する手法を提案した．この手法は，命令長が固定長の命令セットにのみ有効である．

我々は，命令セットにある符号化の自由度を利用した命令セットの多様化手法を提案する．また，その実装案として組込みシステムへの適用を考え FPGA 上に実装を行なった．本手法と Kc らの提案した既存手法との比較を行ない，既存手法と本手法のコスト，性能のオーバヘッドは少ないことがわかった．

2　命令セットの多様化

命令セットアーキテクチャ (ISA) には命令セットとレジスタ等が定義されている．各命令のメモリ表現は命令形式で定められ，命令形式には命令を区別する opcode フィールド等が定義されている．ある 2 つの命令 A と B があったときに opcode フィールドの値を入れ替えることで，命令 A と B のメモリ表現が異なった ISA を作り出すことができる．2 つの ISA は命令 A と B のメモリ表現が異なっているだけで同等の機能を持っている．このような命令の符号化の違いを命令セットのパーソナリティと定義する．我々は，この命令セットのパーソナリティを利用して命令セットを多様化する．

命令セットからどれだけのパーソナリティを生み出せるかを命令セットの自由度と呼ぶ．命令形式によっては，opcode フィールドの入れ替えによるパーソナリティの他に拡張フィールドの入れ替えによるパーソナリティも考えることができる．opcode フィールドと拡張フィールドの入れ替えによって生まれる自由度を命令セット全体の自由度とする．

各命令セット (MIPS32, SH-3, Java VM, 8080) の自由度を評価した．それぞれ MIPS32 は 32 bit 固定長命令で SH-3 は 16 bit の固定長命令，Java VM と 8080 は命令コード 8 bit の可変長命令である．

各命令セットの自由度を表 1 に示す．Java VM の自由度はとても大きく，逆に SH-3 は 8080 よりも少なくなった．命令長が長くて命令数が多ければ自由度も大きくなるとは限らず，命令形式が簡潔であるかどうかが大きく影響している．例えば，一番自由度が多かった Java VM は命令形式が 1 つしかないのに対して SH-3 は 12 個も存在していた．また，opcode フィールドの長さも関係がある．opcode フィールドが長ければ割り当てることのできる命令数が増え，入れ替えが可能な命令数が増える．Java VM では 8 bit の opcode フィールドに対して 201 の命令が定義されており，命令形式が 1 つだけなので自由度は $201!\quad 10^{377}$ となる．

本手法も Kc らの既存手法も換字暗号であるため自由度が多ければ多いほど攻撃に対する耐性も高いといえる．本手法で自由度が低くなった SH-3 は，Kc らのビット転置の手法の自由度 2.63×10^{35} よりも多いため，より攻撃に対して有効性がある．また，Kc らの手法と違い本手法は命令長が可変長の命令セットにも適用が可能であり，Kc ら既存手法と併用が可能である．

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

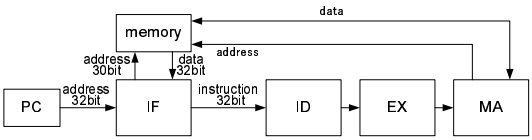

図 1: Plasma のブロック図

3　FPGA による実装

対象とする命令セットとして組込みシステムでよく用いられ，命令形式が簡素で命令長が固定の MIPS32 を選択した．評価環境として，FPGA には Xilinx 社の Spartan3 (デバイス XC3S2000) を使用し，CAD ソフトには Xilinx ISE 8.2i を用いた．また，MIPS 命令セットを実装したソフトコアプロセッサである Plasma [2] を使用した．

Plasma のブロック図を 1 に示す．本手法の実装例として特殊化デザインとメモリマッピングデザインを考えた．特殊化デザインは，オリジナルの命令デコード部分 (図 1 の ID) を書き換える．メモリマッピングデザインでは，命令の写像をメモリに実装し，命令のフェッチ部分 (IF) と ID の間に写像が格納されたメモリを使用して命令を変換する．メモリマッピングデザインの場合は特殊化デザインと違い，写像を格納したメモリを入れ替えるだけでパーソナリティを容易に変更することが可能である．また，既存手法との比較を行なうため Kc の手法 [1] を実装した．ビット転置のデザインは，IF と ID の間に転置されたビット列を元に戻す部分を実装する．鍵を XOR するデザインでは，IF と ID の間に秘密鍵と XOR する部分を実装する．

測定結果を表 2 に示す．性能のオーバヘッドは最大でも 10%前後になり，本手法，既存手法共にオリジナルに対して性能低下は少ないことがわかった．しかし，本手法は Kc ら既存手法よりも自由度が多いためより有効である．

表 2: Plasma の論理合成結果 (Speed 優先)

	Slice (SliceM)	Block RAM	動作周波数 [MHz]	合成時間 [s]
オリジナル	1911 (128)	4	35.3	437
特殊化 (最大値)	2117 (130)	4	37.9	-
特殊化 (最小値)	1868 (128)	4	31.1	-
特殊化 (平均値)	2009 (128)	4	34.1	479
メモリマッピング	1979 (128)	4	31.2	461
ビット転置	2139 (192)	4	33.2	522
鍵を XOR する手法	1887 (128)	4	34.2	426

4　おわりに

命令セットの多様化の手法について提案し，命令セットに含まれる自由度について評価を行なった．本手法は，Kc ら既存手法のものよりも自由度が多く得られることがわかった．

本手法の FPGA への実装を行ない，性能の評価を行なった．本手法も既存手法にもコストと性能へのオーバヘッドは 10%前後と殆どなかったため組込みシステムに対する攻撃への対策として有効な手法であるといえる．本手法と既存手法にオーバヘッドの差は殆どなかったが，自由度の点を考えると本手法は既存手法より有効であるといえる．

本手法は，本質的に換字暗号であるため，頻度分析により解読される恐れがある．また，同一の命令列を多様化したメモリイメージの比較に弱いため，ソフトウェア難読化との併用が望ましい．

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] S. Rhoads: “Plasma - most MIPS I (TM) opcodes: overview,” Nov. 2006. http://www.opencores.org/projects.cgi/web/mips/.