Diversity of Instruction Set Architecture

Graduate Adviser: Shuichi Ichikawa 043714 Takashi Sawada

1 Introduction

The attacks of viruses and worms are serious problems. Viruses often exploit the buffer overflow vulnerabilities to inject a binary code that alters the execution flow of a program (injection attack).

Though there are many preceding studies on protection against injection attack, encryption of memory and verification of instruction sequence incur significant performance overhead.

Another solution is diversification of processors. Kc et al. [1] proposed two methods. One is to XOR a secret key to instruction sequence and the other is to randomly transpose all the bits within the instruction.

This study presents a new method to diversify instruction set architecture by utilizing the redundancy in the instruction set. We evaluated redundancy of four ISAs and Kc's methods. We implemented our methods and Kc's methods by FPGA (Field Programmable Gate Array) chips.

2 Diversity of Instruction Set Architecture

ISA includes definitions of instruction set and register sets. Instructions are defined by instruction formats that consist of various fields. One of these fields is opcode that defines operation. Let us consider a processor P1 that provides ADD instruction (opcode = 1) and SUB instruction (opcode = 2). Here, we can also consider another processor P2, which is a twin of processor P1 except that the respective opcodes of ADD and SUB are 2 and 1 in P2. The difference of P1 and P2 is referred to as *instruction set personality*. It is possible to diversify instruction set architecture by generating instruction set personalities from the original ISA.

The number of personalities that is derivable from an ISA is referred to as redundancy of instruction set. Auxiliary fields for opcode field are also usable to generate new personalities.

In the following evaluation, four ISAs are examined: MIPS a 32-bit architecture, SH-3 a 16-bit architecture, Java VM and 8080 an 8-bit architecture.

The evaluation results of redundancy of each ISA are summarized in Table 1. Java VM involves the largest redundancy, while SH-3 involves the smallest redundancy. A longer instruction format does not necessarily lead to a larger redundancy. Rather, in this study, larger diversity is obtained from architecture with simple instruction formats. For example, the opcodes of Java VM are 1-byte long for any kind of instructions, in which 201 instructions are defined. Consequently, redundancy becomes very large; i.e., 201! 10^{377}

Kc [1] proposed to transpose all the bits randomly within a 32-bit instruction. The redundancy of this method is 32! 10^{35}. Though Kc's methods and ours are both substitution ciphers, our method can yield far larger diversity and is consequently more resistant to brute-force attacks than Kc's methods in any of the four architectures.

Table 1: Redundancy of four instruction sets

	Number of instructions	Redundancy	Information [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

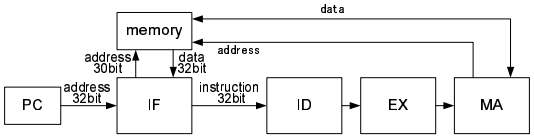

Fig. 1: Block diagram of Plasma

3 Implementation and Evaluation

Our method was implemented and evaluated with Xilinx Spartan3 FPGA technology. As a basis of the following discussion, a soft-core processor Plasma [2], which is based on MIPS instruction set, is adopted. The block diagram of Plasma is illustrated in Figure 1. The following five designs are examined in this study. The Original design is the original Plasma design. Plasma consists of six functional blocks: PC (program counter), IF (instruction fetch), ID (instruction decode), EX (execution), MA (memory access), and memory. Proposed designs are designated as specialized design and RAM-mapped design. The Specialized design implements the ID unit specialized for a processor personality. The RAM-mapped design implements the mapping of opcodes by RAM, consequently enabling changes of personality. The mapping RAM is located between the IF and ID. The Bit-shuffle and XOR design are Kc's methods. The Bit-shuffle design includes a bit-shuffling block, which is simply inserted between ID and IF blocks of the Original design. The XOR design includes a bitwise XOR function with a 32-bit key register between IF and ID blocks.

The evaluation results of implementation of each design are summarized in Table 2. In table 2, Kc's methods and ours show low performance overheads, while our methods involve larger redundancy than Kc's methods.

Table 2: Evaluation results of five designs (optimization option: Speed)

	Slice (SliceM)	Block RAM	Frequency [MHz]	Synthesis [s]
Original	1911 (128)	4	35.3	437
Specialized(Max)	2117 (130)	4	37.9	-
Specialized(Min)	1868 (128)	4	31.1	-
Specialized(Avg.)	2009 (128)	4	34.1	479
RAM-mapped	1979 (128)	4	31.2	461
Bit-shuffle	2139 (192)	4	33.2	522
XOR	1887 (128)	4	34.2	426

4 Conclusion

This paper presents a new method to diversify instruction set architecture by utilizing the redundancy in the instruction set. Our method provides a large degree of freedom in instruction set. Thus, our method is regarded more secure than Kc's methods.

Kc's methods and ours were implemented and evaluated with FPGA technology. Our method is superior to Kc's methods because our methods have larger redundancy than Kc's methods.

Our methods are substitution ciphers. Thus, attacker might exploit frequency analysis in a ciphertext-only attack. Therefore, it is possible and desirable to adopt our method in combination with the preceding methods for more security.

References

[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/.