Design and Evaluation of an FPGA Architecture for Software Protection

Joseph Zambreno

Department of Electrical and Computer Engineering
Northwestern University
Evanston, IL 60208, USA
zambro1@ece.northwestern.edu

Research into defense mechanisms for digital information has intensified in recent years, as malicious attacks on software systems have become a rapidly growing burden. The growing area of *software protection* attempts to address the problems of code understanding and code tampering along with related problems such as authorization. Our work focuses on the design and evaluation of an architecture for software protection that utilizes FPGAs as a run-time integrity enforcement engine.

Recent approaches to software protection tend to lie at two extremes of the security-performance spectrum. At one end are highly secure hardware approaches that require a substantial buy-in from hardware manufacturers [1]. The other end provides security by either mangling the code to make it less understandable or by burying checksums in unlikely places [2]. These extremes invite an approach that allows system designers to position themselves where they choose on the security-performance spectrum.

Our proposed method works as follows. The processor is supplemented with an FPGA-based secure hardware component that is capable of fast decryption and can also recognize strings of keys hidden in plaintext instructions. The FPGA is situated between the highest level of on-chip cache and main memory in order to directly handle the instruction cache miss requests. These instructions would then be translated and verified in some fashion before the FPGA satisfies the cache request. An advantage to this approach is that the compiler's knowledge of program structure allows for tuning of the security and performance of individual applications. Also, the use of FPGAs minimizes additional hardware design and is applicable to a large number of commercial processor platforms. Our initial results [3] demonstrate that the average performance penalty for this approach is not too severe for most applications.

References

1. D. Lie et al. Architectural support for copy and tamper resistant software. In *Proc. of the 9th Int'l Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 168–177, 2000.
2. C. Collberg and C. Thomborson. Watermarking, tamper-proofing, obfuscation: tools for software protection. In *IEEE Transactions on Software Engineering*, pp. 735–746, 2002.
3. J. Zambreno et al. Flexible software protection using hardware/software codesign techniques. In *Proc. of Design, Automation, and Test in Europe*, pp. 636–641, 2004.