Embedded Processor Security

Brian J. d'Auriol, Nguyen Thi Thanh Tuyen, Vo Quoc Hung, Duc Thang,
Hassan Jameel, Le Xuan Hung, S.M.K.R. Raazi, Dao Phuong Thuy,
Ngo Trong Canh, Adil Mehmood Khan, Sunghyun Kim, Shu Lei,
Sakib Pathan, Tran Van Phuong, Sungyoung Lee, Young-Koo Lee
Department of Computer Engineering, Kyung Hee University, Korea
Email: dauriol@acm.org

Abstract ***A preliminary model is introduced in this paper whereby data and its associated security properties are treated as a single atomic unit of information in a hardware-only context. Security-tagged data allows each datum to be properly manipulated with a predictable assurance. This paper addresses the data modeling, process modeling and network modeling of the associated security-tagged data. This work is a part of a larger issue that instruction set architectures (ISAs) do not consider the information assurance implications in its operational environment.***

Keywords: Information assurance, Confidentiality, Integrity, Processor architecture

I. INTRODUCTION

Information assurance typically involves the protection of information. Three common aspects are confidentiality, integrity and availability (often referred to as the CIA triad or just CIA). Confidentiality deals with the appropriateness of how the information is accessed. Integrity deals with the semantic correctness of the information. Availability ensures that the information is accessible at the time it is needed. Many computer security protocols are typically applied in a layering fashion to ensure protection of these aspects of the information.

Security in computer architecture has tended to either be management protective as in the case of privileged (kernel) modes of operation or globally protective as in the case of FIPS 140 [1] compliant architectures. The former is a software-hardware solution whereby an operating system manages the protection by utilizing specialized architecture features. The latter represents a hardware-only approach to security in architectures. Hardware-only approaches tend to be more recent and are motivated by both a concern to better protect encryption/decryption communication processes and by a feeling that the software in the dual-nature solution is subject to compromise and hence protection failures.

The main idea behind the work in this paper is that data and its associated security properties are treated as a single atomic unit of information in a hardware-only context. Security-tagged data allows each datum to be properly manipulated with a predictable assurance in the context of confidentiality, integrity and availability. However, typical architectures are not designed to accommodate the processing of atomic security tagged data nor its memory addressing, data transfer or interprocessor communications. Although tagged data may be logically associated and processed under software control, experience from the hardware-software approaches indicates that doing so is susceptible to potential compromise and protection failures. Applications of security data tagging include screening-out sensitive information from display or removable devices (thereby perhaps preventing similar occurrences of the recent data breach at the U.S. Department of Veterans Affairs [2]), and better control of security over data manipulated by an architecture.

This paper presents preliminary research results aimed at identifying important issues, modeling various aspects of hardware-integrated CIA and assessing the overhead of such integration. In [3], a preliminary model is introduced and assessed by two case studies, one applied in a simple MIPS processor design and the other applied in an optical interconnected multiple processor architecture model. This earlier work concentrated more on the feasibility of the idea; only confidentiality and integrity were considered. This earlier work is ex-

tended in this paper.

The remainder of this paper is organized as follows. The next section, Section II, reviews related work. Section III introduces a preliminary model of security property data tagging and discusses implementation and performance implications of the model. Some comments about the architecture are given in Section IV. Conclusions are given in Section V.

II. Review

Protection of hardware resources has a long history. Multi-tasking and the more recent multi threading operating systems provide management of resources by using privileged, kernel or supervisory modes, for example, BSD [4]. Operating systems may also provide protection over shared memory especially where shared memory is used to support interprocessor communication (IPC) between processes [4]. Such management is made possible by hardware supported features such as kernel accessible memory and kernel-only registers of the MIPS 32 and 34K core [5]. Multiple operating system interfaces can also be supported in a secured way, for example, the privileged architecture library code of the Alpha AXP 21064 [6]. This dual hardware-software approach provides security over process and thread contexts, usually, so that user applications are isolated from each other or can not otherwise interfere with the integrity of the system. However, the well known (and often exploited) limitation of this approach is the security of the operating system software itself. Compromised software leads to the break-down of system security.

Another approach is that of physical hardware security combined with internal hardware security. A FIPS 140 compliant hardware module [1] provides up to four layers of protection. Specified security features include a finite state model describing correct operations, secure key management and, for Level 4, a detection and response "envelope of protection". The IBM 4758 processors are certified under Levels 3 and 4 of the FIPS 140-1 (the previous version of the standard, 1994) [7], [8]. Some of the features implemented in the IBM 4758 are the zeroization of some of the RAM memory, processor subsystem reset, termination of RAM-memory refresh, and a state controller which, amongst other actions, control memory allocation. Detected penetrations result in sensitive data destruction.

There are some hardware-only approaches. Limited information access occurs as a by-product of memory management private data caching in some multiple processor systems [9]. The MIPS 34K core provides a variation of the cache prefetch that can zero out a cache line in certain circumstances [5]. The Cell Broadband Engine (Cell BE) [10] is a recent processor designed with hardware-only support for security, including a "secure processing vault" and instruction stream runtime verification.

The hardware-only approach addresses the limitation of the dual software-hardware approaches. However, the hardware-only approaches surveyed here consider more globally protective mechanisms. In particular, these approaches do not consider data tagged with security attributes specifying allowable operations. Such a model is presented next.

III. Modeling

Based on our preliminary investigations, we divide the issues into the three categories: data modeling, process modeling and network modeling.

A. Data Modeling

Data modeling involves relational modeling of data and security tags. Several possible relationships may be considered. Depending on the situation, both data and security tags may be keyed by their respective memory addresses; or, alternatively, new keys (e.g. hashes of the memory addresses) could be determined (if needed). Insertion, deletion and modification to the data and security tags are necessary operations.

There are two main issues in data modeling. First, the association of security tags with the data. Second, the storage of the data and its security tags. In this paper, we concentrate on the first issue to provide greater understanding about the requirements of the memory architecture needed to adequately support the second. We consider two data models as shown in Figures 1 and 2 and refer to these as Data Model 1 and Data Model 2 respectively.

In Data Model 1, the security attributes are combined with the data items. This provides a physical association of the attributes with the datum.

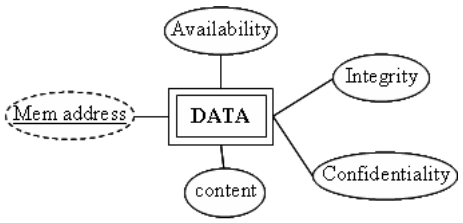

Fig. 1. Data Model 1.

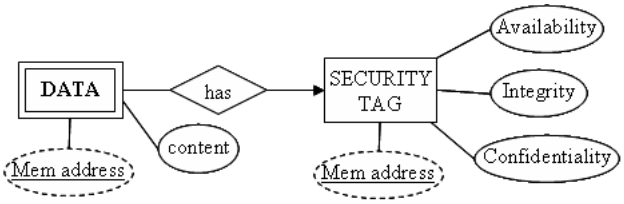

Fig. 2. Data model 2.

In Data Model 2, only the required security tags need be explicitly stored and a datum is associated with a particular tag via the `has` relation; the cross-referencing is based on memory addresses which form the respective information keys. The advantages of this approach include minimal storage requirements of the security tags by reusing tag information for multiple data items. However, the disadvantages include the physical separation of security tags from the data items; in turn, we expect that this would lead to additional issues in a hardware implementation. Moreover, this approach may be extended by including a list of datum memory addresses to which the security attribute belongs to thereby allowing efficient data item selection based on specific attribute queries, but at the cost of additional memory storage and structure maintenance processing times.

B. Process Modeling

Process modeling establishes the necessary requirements needed to consider the implementation of the security attributes. The intention is to identify policy-based requirements within the implementation framework; policies may be conservative (i.e., maximum confidentiality, minimum integrity), averaged (i.e., average confidentiality, average integrity) or others.

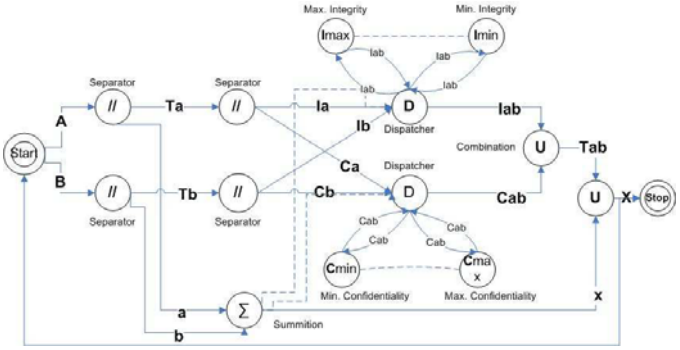

Fig. 3. Process model.

As a high-level CSP description $\bar{P} = (d \rightarrow ((p_1 \rightarrow C)||(p_2 \rightarrow I))$ where $\bar{P}$ is the process of security tag combination, p_1 is the specific process applied to the confidentiality tag C, p_2 the process applied to the Integrity tag I. Figure 3 details this process in which **D** denotes the set of possible operations including maximum, minimum and average. The Summation operation denotes data synthesis, **A** and **B** denote the data items together with security attributes, **Ta** and **Tb** denote the combined security attributes associated with the data items **a** and **b** respectively.

C. Network Modeling

We now consider the communication issues involved in the transmission of data items together with their security attributes. If Data Model 1 is applied to the transmission process, then, each datum has associated tags that need to also be communicated. This increases the TCP payload size. However, if Data Model 2 is applied, then data that share the same tags can be collected together and transmitted with a single copy of the tags. This will increase the payload slightly. To address this, we consider the inclusion of tags into the TCP header but without impacting upon optional requirements, that is, we wish to include the security tags along with whatever other transmission requirements are needed. The TCP header (defined by RFC 793) includes six bits for future use. However, two of these are used for Explicit Congestion Notification (defined in RFC 2481) leaving four bits for use. Since transmission is usually segmented, the security tags can be distributed across multiple segmented headers, for example, two headers can

be used to support tags of four bits each for Confidentiality and Integrity. Including security tags in the header does not impact the payload size, furthermore, allows header processing algorithms to assess the security prior to delivery of payload. The segmentation distribution may also be applied to fewer than the four available bits with a cost of increasing the number of segments needed to support the data transmission; furthermore, bit fields from the urgent points, timestamp, etc. options can be used for specific cases where these fields are not used. A disadvantage beyond the utilization of the expansion bits is that the header is not encrypted, therefore, exposing the security attributes. This may be addressed either by an encryption/decryption protocol prior to packetization or by distributing the security tags around the segments.

IV. Architecture

This section briefly describes the potential architecture implied by the previous discussions.

The two data models impact the nature of the memory layout. Data Model 1 implies that either: the word size is extended to store the tags, the same word size is used and the tags are stored in the higher-order bits, or the same word size is used and the tags are stored in a separate memory word; the latter may either include a wider bus or specific MMU constraints on the required sequence of fetch datum, fetch tags. Data Model 2 implies that the same as the third option above: the same word size is used and the tags are stored in a separate memory word. Again, a wider bus or MMU constraints can govern the integrated fetch (and store) operations. Additionally, since there are a small amount of tags, these could be preloaded into a hardware table-lookup module.

The process model defines a number of security tag combiners. A comparator can be used to select the maximum/minimum tag value. Averaging requires a more advanced circuit; one method is to consider an optimized enhancement to an adder, e.g. [11].

V. Conclusion

A preliminary model is introduced in this paper whereby data and its associated security properties are treated as a single atomic unit of information in a hardware-only context. Security-tagged data allows each datum to be properly manipulated with a predictable assurance. This paper addresses the data modeling, process modeling and network modeling of the associated security-tagged data. This work is a part of a larger issue that instruction set architectures (ISAs) do not consider the information assurance implications in its operational environment.

However, the preliminary work in this paper does not provide realistic implementations. Nor, does it consider applications that may benefit from this approach. Therefore, the work in this paper lays the foundation for both of these future-work investigations.

Acknowledgements

This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITFSIP (IT Foreign Specialist Inviting Program) supervised by the IITA (Institute of Information Technology Advancement).

References

[1] National Institute of Standards and Technology. *Security Requirements for Cryptographic Modules*, May 2001.

[2] Jaikumar Vijayan. Massive data breach puts VA's IT policies under a microscope, May 2006.

[3] Brian J. d'Auriol. Architecture information assurance. In *Proceedings of The First Workshop on Embedded Systems Security, the 6th Annual ACM Conference on Embedded Software (EMSOFT'06)*, Seoul, South Korea, Oct. 2006. published on conference CD.

[4] Marshall Kirk McKusick, Keith Bostic, Michael J. Karels, and John S. Quarterman. *The Design and Implementation of the 4.4 BSD Operating System*. Addison-Wesley Longman, Inc., 1996.

[5] MIPS Technologies, Inc. *Programming the MIPS32 34K Core Family*, September 2005. Revision 01.05.

[6] Richard L. Sites. Alpha AXP architecture. *Communications of the ACM*, 36(2):33–44, February 1993.

[7] IBM. *IBM 4758 Models 2 and 23 PCI Cryptographic Coprocessor*, May 2004.

[8] IBM. *IBM PCI Cryptographic Coprocessor, General Information Manual*, sixth edition, May 2002.

[9] David A. Patterson and John L. Hennessy. *Computer Architecture A Quantitative Approach*. Morgan Kaufmann Publishers, Inc, second edition, 1996.

[10] Kanna Shimizu. The cell broadband engine processor security architecture, April 2006.

[11] Ruby Bei-Loh Lee and John Paul Beck. Parallel adding and averaging circuit and method, 1999. US Patent 4707800 and 4768160.