A Case for Tamper-Resistant and Tamper-Evident Computer Systems

Yan Solihin
Center of Efficient, Secure, and Reliable Computing (CESR)
North Carolina State University
solihin@ncsu.edu

Abstract

Recent industrial efforts in architectural and system support for trusted computing still leave systems wide-open even to relatively simple and inexpensive hardware-based attacks. These attacks attempt to snoop or modify data transfer between various chips in a computer system such as between the processor and memory, and between processors in a multiprocessor interconnect network. Software security protection is completely exposed to these attacks because such transfer is managed by hardware without any cyptographic protection. In this paper, we argue that the threats from such attacks are serious and urgent, and that computer design should place a priority in protection against these attacks.

1 Fundamental limitations of today's security mechanisms

While data transfer between several computer systems that are networked is managed by software, data transfer within a computer system between its components is managed completely by hardware and is transparent to the software. For each computation task, lage amounts of data are transferred between various chips such as the processor and memory, or between processors in a multiprocessor system. Currently, such data transfer is completely unprotected, which can be snooped or altered through relatively simple hardware devices attached to various buses and the interconnects. This presents a serious security challenge in that even the most secure software protection can be broken because its sensitive information is stored as program variables off the processor chip. Furthermore, by snooping data brought into the processor chip, attackers can reverse engineer code, snoop unencrypted data, or even alter data before it enters the processor chip. Recognizing some of these challenges, industrial efforts have resulted in *Trusted Computing* efforts [9, 15]. Unfortunately, Trusted Computing only addresses a small subset of these attacks. While authentication of certain system software is provided with trusted computing, data transfer is still unprotected against snooping and tampering.

Granted, such hardware attacks require the attackers to have physical access to the computer systems, so they are not commonplace yet. However, we believe that there are several important use scenarios of computer systems in which the possibility for such attacks is quite high and needs to be taken very seriously.

The first scenario is when *attackers has almost unlimited physical access to the system* because they either own it, or they administer it. One example from this scenario is consumer electronics such as game consoles and portable media players. Such systems often come with copyright protection mechanism. Users or owners of the system can repeteadly attack the system in order to break such protection mechanism with a strong financial incentive because such devices are common and the cost of designing the attacks can be amortized over many instances. This seriousness of such attacks has been demonstrated by the commercial success of mod-chips, enabled by unencrypted transfer between the BIOS and the processor chip [4].

Another example of such scenario involves *voting machines*. Since these machines are placed in a great number of sites, it is hard to provide them with complete physical security. It is hard to ensure that administrators of the machines will not tamper the machines, or will not unintentionally let others to tamper with them.

Another scenario is when *attackers has limited physical access to the system but there are non-intrusive and traceless ways to attack the system*. Large multiprocessor systems used for utility or on-demand computing servers are particularly vulnerable. In the utility computing model, companies "lease" resources of a large-scale, powerful servers (e.g. the HP Superdome [10]) to customers who need such resources on a temporary basis or who want to offload their IT operations. These large-scale systems are not under the control of the customers who are using their resources. The customers are likely to be wary about adopting the utility computing model unless the secrecy and integrity of their data can be ensured. In fact, concerns about data privacy have been reported to slow down the adoption of utility computing model [1]. If the server system itself does not ensure data confidentiality and integrity, malicious employees or other attackers who can get through the physical security protecting the machine could easily steal or modify important data. The risk of security attacks by selected employees or parties that have physical access to the machine should not be underestimated. For example, in the case of ATMs, Global ATM Security Alliance (GASA) reported that more than 80% of computer-based bank-related frauds involve employees [6]. In the case of DSM systems used for utility computing, the large amounts of sensitive data in these systems create a financial incentive for the attackers to perform corporate espionage or other malicious intents. To make matters worse, such attacks could be performed without disrupting the system, for example by attaching a simple device to an interconnect wire. Such attacks also do not produce traces that can alert other users about the existence of the attacks. These concerns may prompt customers to demand that DSM utility computing systems be equipped with hardware support for data confidentiality before they would be willing to use those systems. This also suggests that data security in DSM systems will become an increasingly important issue in the future.

2 Important research challenges

One main research challenge is how to *efficiently* ensure *privacy*, *tamper-resistant* and *tamper-evident* properties for a computer system. Privacy requires data transfer to be encrypted so that attackers cannot gain much insight into the data from snooping it. Tamper-resistance requires that data transfer is enrcypted in such a way that it is hard for the attackers to tamper the data in a meaningful way. Finally, tamper-evidence requires authentication of data transfer to detect attack attempts and secure logging to record information of the attacks.

Data transfer between chips must be provided with very low latencies, and any delay due to cryptographic operation can significantly slow down the computer systems. For example, current memory access latency is in the order of 200ns, while decryption operation applied to incoming cache block can easily add 30-50% to the latency. Another important challenge is the space overhead due to storing hash codes. In recent studies, to prevent tampering of data transfer, a Merkle tree of hash codes requires a space overhead of 25%. This is clearly unacceptable in a system where performance or cost are critical issues.

Another main research challenge is how to retain the operability of such system. Since the entire memory is encrypted, secure mechanisms are needed in order for the system to communicate with external devices, such the I/O subsystem.

Another major research challenge is how to securely boot the system. For uniprocessor system, this is relatively simple to achieve, but for multiple processors communicating with each other, we need a mechanism to establish trust between the communicating processes. Traditional protocol such as Kerberos is hard to apply because it assumes the existence of secure software. Secure hardware booting cannot assume that the security software is already running.

3 Promising innovations and abstractions for future systems

A body of research exists on memory encryption and authentication schemes for uniprocessor systems [2, 3, 5, 7, 8, 12, 13, 14, 16, 17]. The main assumption in memory encryption and authentication work is that on-chip data is secure and cannot be observed by attackers, while data that resides anywhere off-chip can be observed and altered by attackers using hardware attacks. Therefore, the goal of memory encryption and authentication schemes is to encrypt and hash data before it leaves the processor chip, and then to decrypt and authenticate it when it is brought back on-chip. Several studies use a direct encryption approach where a block cipher such as AES is used to directly encrypt and decrypt data [3, 7, 8]. However, these approaches add the long latency of the block cipher to the critical path latency of off-chip data fetches. To hide this latency, several studies have examined counter-mode encryption where a data block is encrypted or decrypted through an XOR with a pad [12, 14, 16, 17]. The pad is constructed by encrypting a *seed*, which is typically composed of a per-block counter and the block's address. The security of counter-mode encryption relies on uniqueness of pads, which is maintained by by incrementing the block's counter each time the data is updated. Counter-mode hides decryption latency by caching [14, 16, 17] or predicting [12] the block's counter, so pad generation can proceed in parallel with the fetch of the block's data from DRAM. For authentication, Merkle hash trees have been proposed to protect the integrity of data in memory from data tampering and replay attacks. In the Merkle tree scheme, a tree of Message Authentication Codes is formed over the blocks of data in memory, with the root of this tree always kept on-chip. Data integrity can be verified by computing MACs up the tree to the secure root.

Our own research has advanced the state of the art of counter-mode memory encryption and authentication by enabling the processor to hide cryptographic operation latency so that no noticeable slowdown is observed, for both uniprocessor system [16], and large multiprocessor server system [11].

All such technologies serve as a proof-of-concept that efficient memory encryption and authentication can be achieved. However, many research challenges, such as communication mechanism with the external world, secure booting, and tolerating space overheads, remain unaddressed.

4 Possible milestones for the next 5 to 10 years

Milestones should include a working prototype of secure chips. A prototype requires addressing problems that may not be obvious at the research stage, such as the impact of the design on the Operating System and application software. It is also useful to subject the prototype to various attacks on data transfer to make sure that the protection is reasonably secure and securely implemented. Finally, prototyping requires the changes to existing systems to be reduced to a minimum while still providing strong security.

References

[1] D. Bartholomew. On Demand Computing – IT On Tap? *http://www.industryweek.com/ReadArticle.aspx?ArticleID=10303 &SectionID=4*, June 2005.

[2] B. Gassend, G. Suh, D. Clarke, M. Dijk, and S. Devadas. Caches and Hash Trees for Efficient Memory Integrity Verification. In *Proc of the 9th Intl. Symp. on High Performance Computer Architecture (HPCA-9)*, 2003.

[3] T. Gilmont, J.-D. Legat, and J.-J. Quisquater. Enhancing the Security in the Memory Management Unit. In *Proc. of the 25th EuroMicro Conf.*, 1999.

[4] http://www.modchip.com, 2005.

[5] IBM. IBM Extends Enhanced Data Security to Consumer Electronics Products. *http://domino.research.ibm.com/comm/pr.nsf/pages/ news.20060410_security.html*, April 2006.

[6] M. Lee. Global ATM Security Alliance focuses on insider fraud. *ATMMarketplace, http://www.atmmarketplace.com/article.php? id=7154*, 2006.

[7] D. Lie, J. Mitchell, C. Thekkath, and M. Horowitz. Specifying and Verifying Hardware for Tamper-Resistant Software. In *IEEE Symp. on Security and Privacy*, 2003.

[8] D. Lie, C. Thekkath, M. Mitchell, P. Lincoln, D. Boneh, J. MItchell, and M. Horowitz. Architectural Support for Copy and Tamper Resistant Software. In *Proc. of the 9th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, 2000.

[9] Microsoft Corporation. Microsoft Next-Generation Secure Computing Base – Technical FAQ. http://www.microsoft.com/technet/archive/security/news/ngscb.mspx, 2003.

[10] T. Olavsrud. HP Issues Battle Cry in High-End Unix Server Market. *ServerWatch, http://www.serverwatch.com/news/article.php/ 1399451*, 2000.

[11] B. Rogers, Y. Solihin, and M. Prvulovic. Efficient data protection for distributed shared memory multiprocessors. In *Intl. Conf. on Parallel Architectures and Compilation Techniques*, 2006.

[12] W. Shi, H.-H. Lee, M. Ghosh, C. Lu, and A. Boldyreva. High Efficiency Counter Mode Security Architecture via Prediction and Precomputation. In *32nd Intl. Symp. on Computer Architecture*, 2005.

[13] W. Shi, H.-H. Lee, C. Lu, and M. Ghosh. Towards the Issues in Architectural Support for Protection of Software Execution. In *Workshop on Architectureal Support for Security and Anti-virus*, pages 1–10, 2004.

[14] G. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas. Efficient Memory Integrity Verification and Encryption for Secure Processor. In *Proc. of the 36th Intl. Symp. on Microarchitecture*, 2003.

[15] Trusted Computing Group. https://www.trustedcomputinggroup.org, 2005.

[16] C. Yan, B. Rogers, D. Englender, Y. Solihin, and M. Prvulovic. Improving cost, performance, and security of memory encryption and authentication. In *Proc. of the Intl. Symp. on Computer Architecture*, 2006.

[17] J. Yang, Y. Zhang, and L. Gao. Fast Secure Processor for Inhibiting Software Piracy and Tampering. In *Proc. of the 36th Intl. Symp. on Microarchitecture*, 2003.