Secure Processing On-Chip

Hsien-Hsin S. Lee

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250
leehs@ece.gatech.edu

Santosh Pande

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
santosh@cc.gatech.edu

ABSTRACT

Providing security in embedded systems is in urgent needs while there are many challenges in both software and hardware sides that require further research to understand their implications. This paper discusses microarchitectural and compiler support to address a variety of vulnerabilities due to physical tampering, program behavior exploits, and digital rights management issues. We also advocate the need for protecting intellectual properties programmed in the growing number of FPGA-based embedded systems.

1. INTRODUCTION

While embedded computing is becoming more pervasive and invisible, the ways users communicate and operate data on these devices, however, are becoming more vulnerable to malicious exploits. When these data, either sensitive or insensitive, are manipulated in a way they are not intended for, some dire consequence may ensue. For example, crackers can reverse-engineer the cryptographic keys of a multimedia system or game console to duplicate and distribute illegal copies of proprietary software [1]. Another example described in [2] shows that well-resourced crackers can invade one's privacy by monitoring the thermostat to determine if one is at home or not. Even worse, malicious attackers can change the setting of the thermostat through Internet and damage pipes or kill pets during winter times.

To provide reliable security for these devices to combat against various types of attacks remain a major challenge to both hardware and software designers. The reality is that embedded system designers can no longer consider security as an afterthought as many robust security features require shrewd and thorough consideration at the very early design stage. In this paper, we discuss potential security breach from a system's perspective at the microarchitecture level and the compiler level. We hope our advocates will raise the consciousness of building security as an indispensable part in the embedded system design flow.

2. PHYSICAL TAMPERING

One of the greatest concerns on embedded devices is regarding malicious exploits via physical tampering of the devices when adversaries gain full physical access to the hardware and reveal sensitive data or intellectual property (IP) algorithms employed in these compromised devices. Obviously, secrets inside these devices must be protected against these physical attacks. However, the issue becomes even more challenging when both IP protection requirement and real-time constraint need to be met for these embedded applications. To guarantee both criteria, hardware-based encryption support is generally implemented to provide satisfactory performance while caution must be made to not adding too much cost to the systems. Nevertheless, employing encryption alone is not sufficient to avoid new types of attacks via other new breed of attacks such as

using side channels [3, 4, 5]. Vulnerability can be exploited by analyzing information leaked through these channels. For example, the absolute and relative locations of the program code are not altered during instruction fetch. In other words, addresses are issued on the bus as plaintext and can be probed by crackers to reconstruct the control-flow graph of a program. Such a vulnerability is particularly pronounced in embedded processors, which typically do not employ cache hierarchies for the requirement of predictable timing. Even with the presence of an instruction cache, a cracker can still easily circumvent the cache by turning off the cache or flushing the cache to force instruction addresses shown on the external bus. In some cases, such information leakage can lead to the revelation of critical information such as encryption keys or passwords of the compromised systems. Another example of the same type of exploits is differential power analysis (or DPA). As shown in previous studies, a well-equipped and motivated cracker can perform non-invasive power (or current) analysis by using oscilloscope on an embedded device such as Smart Cards to retrieve secrets. The idea is based on the observation that power dissipation is strongly correlated to different program behavior on a processor, which can then be used as a signature to compromise secrets. Furthermore, the growing application of low-power techniques such as clock gating makes such attacks even easier.

To combat such issues, effective and efficient obfuscation techniques must be considered, in particular, building them directly into the hardware at the microarchitectural and circuit levels. Fundamentally, obfuscation is aimed to randomize any trace or signature exhibited from address stream or measurable power or current consumption, making distinctive computation operations indistinguishable. A solution demonstrated by [6] uses an on-chip shuffle buffer to perform randomization for the address footprint. The shuffle buffer, essentially an extended small memory array but exclusive to the memory, was designed to reorder all addresses to the memory, obfuscating the address recurrences. Addresses that are ready to be evicted from the shuffle buffer due to a conflict will swap their locations between the shuffle buffer and the main memory. As such, the same address request will appear differently on the bus every time and the goal to evenly distributing the observed addresses can be achieved. Several other literature [7, 8] also investigated such address leakage issues for different system platforms.

3. CONTROL FLOW VULNERABILITY

Exploits such as buffer overruns that alter the program behavior by injecting malicious codes or manipulating high-privileged users inputs represent another major concern. The latter often interacts with input channels such as keyboard or network connection and changes the intended program flow to accomplish their illegitimate actions. Note that a pure software countermeasure can be slow and incapable of detecting such violation. To make the software more robust and evident to such attacks, anomaly detection mechanisms

need to be established. An anomaly system is aimed to monitor program execution and raise an alarm whenever there is a detected abnormal program behavior such as program is redirected to unintended or undefined program paths.

An effective mechanism requires to enforce the control-flow awareness via compiler's analysis and microarchitectural support to enable the protection with high efficiency and high accuracy. For instance, an Infeasible Path Detection System (IPDS) proposed in [9] explores the synergy of compiler and microarchitecture to counteract such memory tampering attacks causing invalid program control flow. In the proposed system, the compiler analyzes correlations among conditional branches to realize illegal program flow changes. Then the collected information is made available to the runtime system. The runtime system, with the support of small hardware tables, will detect dynamic violation of infeasible program paths based on the static information.

4. DIGITAL RIGHTS MANAGEMENT

With the emergence of online commerce on virtual properties such as 3D game characters or arts, to protect these intellectual property on embedded devices and to restrict their usage have become a new design challenge. The recent incident of hacking Xbox [1] furthers the urgent need to include native hardware support for providing a more robust digital rights management (DRM) to enable a tamper-proof embedded platform. To integrate such protection scheme into media processing systems more seamlessly and securely without compromising performance, it requires that security experts and embedded hardware and software designers to align their tasks together. A DRM-enabled 3D graphics processor was demonstrated in [10]. It consists of two components, a cryptographic unit that decrypts protected IP data, and a license verification unit that authenticates the license of these data. Similar to digital rights licenses used in other content protection scenarios, the graphics digital rights licenses released by their providers specify and designate the desired usage of the graphics data. Under this system, exploits are prevented by restricting the otherwise arbitrary bindings among geometry input, textures and shaders through the licenses that define the legal bindings of these objects. During rendering, the binding context consisting of decryption keys and digests of protected data will be checked and verified in the cryptographic hardware units. Additionally, such a DRM-enabled graphics system also protects the Z-buffer, i.e. the depth information, to prevent crackers from reconstructing a 3D geometry model by dumping out the Z-buffer values.

5. IMPLICATIONS OF FPGA-BASED DESIGN

More recently, due to the substantial improvement in FPGA technology, digital designs using FPGA is no longer simply for early prototype or proof-of-concept. In fact, products are being implemented using FPGA for its efficiency (design turnaround time), reconfigurability, and flexibility. FPGA is also an attractive solution for implementing cryptographic applications to adapt the needed changes and enhancements in security policies. An example is set-top boxes which use FPGA to encrypt and decrypt the media stream for pay-per-view movies. Even though the above applications seem to fall into two different groups, yet their demands in security are almost identical — i.e., how to protect the contents implemented and configured in the FPGA? The contents from the first category are related to the IP (i.e. the algorithms) issues of a proprietary design, while the contents from the second category will contain critical secrets such as the cryptographic keys. Similar to what we described earlier, FPGA-based designs suffer from physical tampering — from IP theft by simply reading bitstream out of

the FPGA to DPA side-channel attacks. To address such vulnerabilities, new ideas are needed for both FPGA chip vendors and synthesis tools and algorithms to protect the contents programmed on the gate arrays.

6. CONCLUSION

We are entering an interesting time for embedded designers to (re)consider security as a top design priority at the early design stage. The problem is multi-faceted, involving all layers in a design including the system software (OS and compiler), architecture, microarchitecture, and circuits. Several challenges are lying ahead and a holistic solution across the stack is in need.

In this paper, we are advocating to integrate inherently high security hardware and system support to embedded processors. These schemes typically require dedication of on-chip hardware resources being used to achieve high efficiency and be effective. Nevertheless, any additional hardware feature for cost-constrained embedded systems must be carefully evaluated and justified. Another challenge of integrating security solutions in embedded systems is power consumption, which is already a constraint for battery-powered devices. It will become worse when obfuscation techniques are applied to randomize and disguise program behavior. Adding security to both compiler and hardware levels could also procrastinate the design turnaround time, a critical cost and competitiveness concerns given the short time-to-market cycles of these products. All these trade-offs need to be deliberately balanced in the design of future embedded systems to enable highly security processing.

7. REFERENCES

[1] Andrew Huang. *Hacking the Xbox: An Introduction to Reverse Engineering*. No Starch Press, 2003.

[2] Philip Koopman. Embedded System Security. *IEEE Computer*, pages 95 – 97, July 2004.

[3] P.C. Kocher, J. Jaffe, and Jun B. Differential Power Analysis. In *Proceedings of Advances in Cryptology, Crypto 1999*, 1999.

[4] Weidong Shi, Hsien-Hsin S. Lee, Chenghuai Lu, and Mrinmoy Ghosh. Towards the Issues in Architectural Support for Protection of Software Execution. *SIGARCH Computer Architecture News*, 33(1):6–15, 2005.

[5] Weidong Shi and Hsien-Hsin S. Lee. Authentication Control Point and its Implications for Secure Processor Design. In *Proceedings of the 39th Annual International Symposium on Microarchitecture*, pages 103–112, 2006.

[6] Xiaotong Zhuang, Tao Zhang, Hsien-Hsin S. Lee, and Santosh Pande. Hardware Assisted Control Flow Obfuscation for Embedded Processors. In *Proceedings of the 2004 International Conference on Compilers, Architectures, Synthesis on Embedded Systems*, pages 292–302, Washington D.C., 2004.

[7] Lan Gao, Jun Yang, Marek Crobak, Youtao Zhang, San Nguyen, and Hsien-Hsin S. Lee. A Low-Cost Memory Remapping Scheme for Address Bus Protection. In *In Proceedings of International Conference on Parallel Architectures and Compilation Techniques*, pages 74–83, September 2006.

[8] Xiaotong Zhuang, Tao Zhang, and Santosh Pande. HIDE: An Infrastructure for Efficiently Protecting Information Leakage on the Address Bus. In *the 11th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 72–84, 2004.

[9] Xiaotong Zhuang, Tao Zhang, and Santosh Pande. A Low-Cost Memory Remapping Scheme for Address Bus Protection. In *In Proceedings of the 39th International Symposium on Microarchitecture*, pages 113–122, September 2006.

[10] Weidong Shi, Hsien-Hsin S. Lee, Richard M. Yoo, and Alexandra Boldyreva. A Digital Rights Enabled Graphics Processing System. In *In Proceedings of the ACM SIGGRAPH/Eurographics Workshop of Graphics Hardware*, pages 17–26, 2006.