Trusted Hardware: Can It Be Trustworthy?

Cynthia E. Irvine
Department of Computer Science
Naval Postgraduate School
Monterey, California
irvine@nps.edu

Karl Levitt
Program Director, CISE/CNS
National Science Foundation
4201 Wilson Blvd.
Arlington, Virginia 22230
klevitt@nsf.gov

ABSTRACT

Processing and storage of confidential or critical information is an every day occurrence in computing systems. The trustworthiness of computing devices has become an important consideration during hardware design and fabrication. For instance, devices are increasingly required to store confidential information. This includes data such as cryptographic keys, personal information, and the intellectual property (IP) in the device's design. Furthermore, computing systems in critical applications must work as specified. Therefore it is important that hardware be designed and fabricated to be trustworthy.

Many potential attacks can be used to exploit a computing device. Physical attacks, that monitor power, timing, electromagnetic radiation, etc. can be used to steal confidential information from the system. A "malicious" foundry can perform a number of devious activities including stealing the mask, reverse engineering IP, subverting the hardware through back doors and time bombs, and overproducing counterfeit chips. Design tools can be subverted to insert malicious circuitry, and chip packagers can modify selected devices with their own that provide similar functionality, in addition to underhanded behavior, e.g. stealing information or malfunctioning at critical junctures.

The notions of trust and trustworthiness are presented. Although major challenges still confront secure software system development, there has been substantial progress.Techniques that have been useful in the context of software systems are described and their relevance to the hardware domain is discussed. Challenges to trusted hardware development are then explored.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: [Access controls, Information flow controls, Verification]; B.7.1 [**Types and Design Styles**]: [Algorithms implemented in hardware, Gate Arrays, Microprocessors and microcomputers]

General Terms

Security, Hardware

Keywords

Trust, Assurance, Processor, FPGA, ASIC, Threats, Vulnerabilities, Evaluation

1. A BASIS FOR TRUST

Although "trusted" is a term of art within the computer security community[1], the way systems are used on a daily basis in rest of the world is quite different. Any system that the end user places confidence in, whether warranted or not, is trusted. Hence we need precise terminology.

> **trusted:** the degree to which the user or a component depends on the trustworthiness of another component. For example, component A trusts component B, or component B is trusted by component A. Trust and trustworthiness are assumed to be measured on the same scale.[2]

Trustworthy describes components that merit our trust.

> **trustworthy:** the degree to which the security behavior of the component is demonstrably compliant with its stated functionality (e.g., *trustworthy component*).[2]

How much trustworthiness is required in a particular system or component? The answer depends upon how and where that component or system is to be used, i.e., how much protection is needed for the information assets that it will process?

1.1 Threats and Vulnerabilities

Confidence that a system is sufficiently trustworthy to be trusted with sensitive, critical information is achieved by reducing its vulnerability to attach throughout the system life cycle. Threats, which may be measured relative to the value of the information, must be balanced against the cost of vulnerability mitigation. If no potential attacks are envisioned, then there is little risk to information in a system

[1]Traditionally, a trusted system or application is one that enforces security policy, supports its enforcement, or executes within the perimeter of security-relevant functions and adheres to the intent of the security policy, all with a measurable level of confidence.

fraught with vulnerabilities. In contrast, a system containing very valuable information may be juicy prey for highly sophisticated adversaries, and its vulnerabilities must be reduced or removed so that the cost to the attacker becomes prohibitive. Threats range from a malicious insider on the development team to a bad agent who performs maintenance on a fielded system.

The cost of error in hardware can be very high if millions of devices must be recalled to address a flaw in its design or implementation. Lacking the option for the unfortunate, but ubiquitous, downloadable software patch, hardware developers must pay attention to functional aspects of their devices. How might techniques used to avoid flaws be used to increase trustworthiness?

Since protection is not free, it would be wasteful to incur great expense to construct a highly secure system only to protect worthless information assets. The effort expended to address security threats must be balanced against the damage that would result from information exposure or corruption. Two sets of threats must be addressed in the context of secure systems: developmental threats and operational threats.

Developmental threats are those that result in the incorrect construction of the system, viz., the concrete system implementation does not reflect the intent of the high level policy and specifications. This can result from unintentional errors, or, more troubling, the intentional insertion of unspecified functionality in the form of trap doors [13][25]. A rigorous development methodology can mitigate such threats, and can be applied to people, processes, and tools.

Operational threats can expose information assets to a variety of attacks. If the system is incorrectly specified, then a well-constructed interface might be legitimately used in an unanticipated manner for nefarious purposes. A system that is poorly designed and implemented may contain exploitable flaws. Sophisticated attackers can construct beaded attacks that ultimately lead to the "keys to the kingdom". Sometimes the system may be constructed such that its operational state can be manipulated in ways that exfiltrate protected information. These are covert channels [16]: they can be both difficult to eliminate and difficult to exploit. In the case of hardware, exploitation make take the form of side channel attacks, e.g. [15], that extract information using power or timing analysis.

1.2 Assurance and Lifecycle Considerations

Because attackers will choose the path of least resistance to realize their objectives, achieving trustworthy systems is an assurance life cycle challenge that must address trust from cradle to grave. Life cycle support must define the environment, tools and techniques to be used, and must include development processes as well as flaw remediation. A configuration management plan and system must be in place to guard against unauthorized modification of the system, as well as ensure the integrity of all tools used for specification, design and development. Development may include a sequence of specification, refinement, and mapping to ensure a correctly implemented system. When warranted, development may entail formal methods. Testing will ensure that the system meets its functional specification, while vulnerability assessment permits the analysis, albeit incomplete, of the system's robustness against exploitation. Plans and procedures must be in place to ensure the safe delivery of the system to its users, and the users need appropriate guidance and documentation so that they can configure and run the system without introducing vulnerabilities.

The assurance techniques described above add time and expense to product development. In an era when corporations must maximize profits and rapidly update products with new features, how are customers to know whether the components they acquire have really been created in a manner that addresses these life cycle concerns? It might be foolhardy to believe vendors without some form of third party assessment. Over the years software and system assurance frameworks to provide guidance to developers and confidence to customers have been developed, e.g. the TCSEC [5] and the current Common Criteria (CC) [12]. The CC describes ten types of assessment required to determine if a system is trustworthy:

1. analysis and checking of processes and procedures;
2. checks that processes and procedures are applied;
3. analysis of the correspondence between design representations;
4. analysis of the design representation against the requirements;
5. verification of proofs;
6. analysis of guidance documents;
7. analysis of the functional tests developed and the results provided;
8. independent functional testing;
9. vulnerability analysis (including flaw hypothesis [19]);
10. penetration testing.

The new Protection Profile for Separation Kernels in Environments Requiring High Robustness [11] levies many of the above requirements on developers who incorporate custom hardware in their systems, whereas the requirements for commercial-off-the-self (COTS) platform components call for the developer to show that those components are capable of effectively supporting the intended system's security functions. Even so, developers can only hope that the COTS components were developed using techniques to achieve trustworthy results. What sort of guarantees beyond saying "trust me" will hardware vendors provide their customers?

2. TRUSTED HARDWARE CHALLENGES

Do the threats and vulnerabilities associated with hardware directly parallel those that have been studied for software systems or does hardware eliminate some threats and vulnerabilities while introducing others? Subversion is perhaps the most ominous threat to software systems and can be accomplished relatively easily [1]. Is the same true for hardware?

Can hardware become more trustworthy by adopting the assurance techniques used for software? What are the similarities between hardware and software development and what translation is required to map to hardware development assurance processes and techniques similar to those applied to software systems? Does the hardware design and implementation process entail techniques with no counterparts in software? If so, how will the requirements for new hardware assurance methods be articulated and met? These are among the issues that need to be examined.

To explore the context for some of these questions, we examine recent progress in three areas: the design and implementation of trustworthy hardware, the kinds of hardware used to enhance platform security, and security applications for which hardware is well suited.

2.1 Construction of Trustworthy Hardware

Where covert channels are a concern, i.e., where mandatory security policies must be enforced, care must be exercised to avoid them at all levels of the protection system. In modern processors techniques to enhance performance, such as speculative execution, can introduce covert channels [27]. Various formal frameworks, e.g. PVS [23] and ACL2 [14], can be used to verify hardware and to analyze assumptions made regarding security properties and secure state.

Disconnects in semantics as well as simple errors can occur between the mathematical description and the concrete implementation of cryptography. These problems are exacerbated in hardware implementations where gulf between the mathematics and the implementation is huge. Data model uniformity can be exploited to develop domain-specific languages that can assist mathematicians and engineers to perform test and verification of the cryptography-relevant elements of systems, permit unambiguous specification of cryptographic algorithms, and enhance reusability on different hardware targets [6].

On FPGAs, mechanisms can monitor for irregular activity and to provide resilience against denial of service attacks [7]. Another form of on-board security mechanism is a *reference monitor* that adjudicates access to memory according to policy [10]. Through a *policy compiler*, users can describe a policy which is converted into a circuit that may be loaded onto the FPGA. Isolation and controlled entry-points are important techniques for protection in any system. In the case of FPGAs, this may be achieved through spatial separation, constrained interconnects between cores; and selective partial reconfiguration with object reuse security [9].

2.2 Types of Trustworthy Hardware

Cryptographic co-processors can range from NICs with built-in cryptographic algorithms to those such as the IBM 4758 which was designed to detect and respond to tampering [22]. The Trusted Computing Group Trusted Platform Module provides cryptographic primitives and protected storage locations that may be used to securely attest aspects of the system configuration to remote parties [26]. The protection of cryptographic data has been addressed through the extension of traditional hardware with a *Secret Protected* unit [17], which stores and protects system keys and a hash value to provide tamper evidence of user keys. Furthermore, the integrity of software can be ensured through the use of protected checksums that are validated on instruction execution.

Hardware mechanisms are used as the foundation for many highly secure systems. Traditional desirable features include: distinct execution entities, distinct address spaces, built-in access checks, multiple processor modes, mechanisms for transfer of control, support for partitioning of task modules, I/O device support, and privileged instructions. All of the highest assurance products evaluated under the TCSEC were based on platforms from the X86 family [21], which offers many of these features. Perhaps one of the most interesting processors with the potential to support secure systems was the Intel 432 [18]. With tight hardware/software coupling and the treatment of everything as a typed object for which programmers could create type mangers, this processor provided an example of tagging and capabilities. The use of such constructs to build secure systems remains a topic of interest and discussion.

Currently vendors are developing next-generation platforms with enhancements that go well beyond those of traditional processors. These processors support virtualization and a variety of other features such as mechanisms for protected storage across power cycles, support to identify and verify the current platform configuration as well as to identify the platform itself [8][29].

2.3 Applications of Secure Hardware

That trustworthy hardware would welcomed as the basis for constructing secure systems ranging from hand-held devices to large servers is a *sine qua non.* There are a number of other applications for which trusted hardware is appropriate.

FPGAs may be used to support security objectives with a variety of content processing solutions such as intrusion detection, XML review, and virus scanning. Consider intrusion detection. Because granular intrusion detection is a computationally intensive task that requires high throughput and employs algorithms that can be encoded using integer operations, it is a good candidate for hardware implementation. The reconfigurability of FPGAs allows modification to parameters so that intrusion detection systems can adapt to new attacks. Examples include string matching engines [24] [4], and the use of principle component analysis [20],

FPGA technology may augment the processor core to mitigate threats of software piracy and software tampering. Zambreno et al. have described an architecture that offers software tamper protection in combination with techniques to thwart observation of software ranging from simple obfuscation to full encryption [28].

3. FUTURE DIRECTIONS

The use of FPGAs in support of high performance reconfigurable computers is becoming increasingly attractive as a result of enormous improvements in performance, size, power consumption, and cost [3]. Through the use of specialized development tools, a high performance platforms can be tailored to certain classes of applications. These systems will provide a rich context in which a variety of techniques to ensure that intended security policies are enforced.

Hardware is the foundation upon which all systems must be constructed. Today, more often than not, system developers treat their hardware as axiomatic. Unfortunately the provenance of most components is such that any claims regarding their trustworthiness are largely unfounded. With increased reliance on sophisticated hardware components, many that perform security-critical functions and some of which are mutable, entire infrastructures may be vulnerable to attack. Through new initiatives and ongoing research programs, it is possible to anticipate energetic efforts to establish principles, tools, and techniques for the design and construction of trustworthy hardware.

4. ACKNOWLEDGMENTS

A portion of this material is based upon work supported by the NSF under Grant No. 0524707. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

5. REFERENCES

[1] E. A. Anderson, C. E. Irvine, and R. R. Schell. Subversion as a threat in information warfare. *Journal of Information Warfare*, 3(2):52–65, 2004.

[2] T. V. Benzel, C. E. Irvine, T. E. Levin, G. Bhaskara, T. D. Nguyen, and P. C. Clark. Design principles for security. ISI-TR-605, Information Sciences Institute, Santa Monica, California, and NPS-CS-05-010, Naval Postgraduate School, Monterey, California, 2005.

[3] D. Buell, T. El-Ghazawi, K. Gaj, and V. Kindratenko. High performance reconfigurable computing. *IEEE Computer*, 40(3):23–27, March 2007.

[4] Y. H. Cho and W. Mangione-Smith. Deep packet filter with dedicated logic and read only memories. In *IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 125–134, 2004.

[5] DoD. *Department of Defense Trusted Computer System Evaluation Criteria*. Number DoD 5200.28-STD. National Computer Security Center, December 1985.

[6] Galois Connection, Inc. *Cryptol Reference Manual*. Galois Connection, Inc., Beaverton, OR, February 2006.

[7] G. Gogniat, T. Wolf, and W. Burleson. Reconfigurable security architecture for embedded systems. In *Mobile Computer Hardware Architectures: Design and Implementation*, January 2006.

[8] D. Grawrock. *The Intel Safer Computing Initiative*. Intel Press, Hillsboro, OR, 2006.

[9] T. Huffmire, B. Brotherton, G. Wang, T. Sherwood, R. Kastner, T. Levin, T. Nguyen, and C. Irvine. Moats and drawbridges: An isolation primitive for reconfigurable hardware based systems. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2007 (to appear).

[10] T. Huffmire, S. Prasad, T. Sherwood, and R. Kastner. Policy-driven memory protection for reconfigurable hardware. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, volume LNCS 4189, pages 461–478, Hamburg, Germany, September 2006. Springer.

[11] IAD. *U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness*. National Information Assurance Partnership, version 1.021 edition, March 2007.

[12] ISO/IEC. *ISO/IEC 15408 - Common Criteria for Information Technology Security Evaluation*. Number CCIMB-2005-08-001, CCIMB-2005-08-002, CCIMB-2005-08-003. International Organization for Standardisation, version 2.3 edition, August 2005.

[13] P. A. Karger and R. R. Schell. Multics security evaluation: Vulnerability analysis. Technical Report ESD-TR-74-193, Vol. II, Information Systems Technology Application Office Deputy for Command and Management Systems Electronic Systems Division (AFSC), Hanscom AFB, Bedford, MA 01730, 1974.

[14] M. Kaufmann and J. S. Moore. ACL2 homepage. http://www.cs.utexas.edu/users/moore/acl2/.

[15] P. C. Kocher. *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, pages 104–113. Advances in Cryptology – CRYPTO'96, Lecture Notes in Computer Science, V. 1109. Springer-Verlag, 1996.

[16] B. W. Lampson. A note on the confinement problem. *Communications of the A.C.M.*, 16(10):613–615, 1973.

[17] R. Lee, P. Kwan, J. McGregor, J. Dowskin, and Z. Wang. Architecture for protecting critical secrets in microprocessors. In *Proceedings 32nd International Symposium on Computer Architecture*, pages 2–13, Madison, Wisconsin, June 2005. IEEE Computer Society.

[18] H. M. Levy. *Capability-based Computer Systems*. Digital Press, Bedford, MA, 1984.

[19] R. R. Linde. Operating system penetration. In *National Computer Conference*, pages 361–367, 1975.

[20] D. Nguyen, A. Das, G. Memik, and A. Choudhary. A reconfigurable architecture for network intrusion detection using principal component analysis. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Napa Valley, California, April 2006.

[21] O. Sibert, P. A. Porras, and R. Lindell. The Intel 80x86 processor architecture: Pitfalls for secure systems. In *Proceedings 1995 IEEE Symposium on Security and Privacy*, pages 211–222, Oakland, CA, May 1995. IEEE Computer Society Press.

[22] S. Smith and S. Weingart. Building a high-performance, programmable secure coprocessor. *Computer Networks*, 31:831–860, November 1999.

[23] SRI International. PVS specification and verification system. http://pvs.csl.sri.com.

[24] L. Tan, B. Brotherton, and T. Sherwood. Bit-split string-matching engines for intrusion detection and prevention. *ACM Transactions on Architecture and Code Optimization*, 3(1):3–34, March 2006.

[25] K. Thompson. Reflections on Trusting Trust. *Communications of the A.C.M.*, 27(8):761–763, 1984.

[26] Trusted Computing Group. TPM Main, Part 1, Design Principles. https://www.trustedcomputinggroup.org/downloads /specifications/tpm/tpm, 29 March 2006.

[27] Z. Wang and R. B. Lee. Covert and side channels due to processor architecture. In *Proceedings of the 22nd Annual Computer Security Applications Conference*, pages 473–482, December 2006.

[28] J. Zambreno, D. Honbo, A. Choudhary, R. Simha, and B. Narahari. High-performance software protection using reconfigurable architectures. *Proceedings of the IEEE*, 94(2):1–13, February 2006.

[29] A. Zelchick. Processor-based virtualization, AMD64 style, part i. http://developer.amd.com/articles.jsp?id=14&num=1, 2007.