Educating Engineers to Design Trustworthy Systems

Shiu-Kai Chin
Department of Electrical Engineering and Computer Science
Syracuse University, Syracuse, New York 13244
http://lcs.syr.edu/faculty/chin

Abstract

Cyberspace cannot exist without computer hardware, software, and protocols. Forty years of progress has moved us from a thousand transistors to a half-billion transistors per chip. We have moved from single mainframe computers to global connectivity. Safety, security, and trust in cyberspace cannot exist without trustworthy computer hardware, software, and protocols. Yet, the design principles and methods that lead to trustworthiness remain neglected. This neglect exists in large part because undergraduate students are not taught how to design with security and trustworthiness in mind. This paper addresses what could be done differently to educate undergraduate computer engineering and computer science students to meet this growing need.

Keywords: Education, security, trustworthiness

I. Introduction

> *The definition of insanity is doing the same thing over and over and expecting different results*[1].

The education of undergraduate computer engineers and computer scientists has not kept pace with the need for security and trustworthiness. Security pundits routinely call for security to be built into systems from the start. However, the lack of education in rigorously relating security requirements to designs results in engineers and computer scientists graduating without any capability to meet this need. Given that the next generation of engineers and computer scientists designs the next generation of computer systems, why would we expect the security and trustworthiness of future systems to improve without a corresponding improvement in education?

Educators correctly point out that the undergraduate curriculum and courses are already full. This has always been the case. Nevertheless, curricular innovations are made to meet critical needs. One example is the introduction of VLSI (very large scale integrated) circuit design into the undergraduate curriculum. In the 1970's, VLSI design was thought to be too exotic and complex to be taught at the undergraduate level. Engineers of the time routinely spoke of the need for "tall thin men," i.e., engineers who could translate algorithms into working silicon by relating algorithmic specifications to register-transfer level designs implemented as custom VLSI circuits. People of the time lamented the lack of VLSI designers in professional practice. Comments such as, "there are fewer VLSI designers than there are NFL (National Football League) players" were common.

VLSI design is now routinely learned by undergraduate electrical and computer engineers. We know now that enough talented electrical and computer engineers were educated to create Silicon Valley in the US and the global semiconductor industry in Asia, all of which led to the Internet as we know it today.

What can we do differently to achieve a similar revolution for security and trustworthiness? This paper attempts to answer this question. It is based on our experiences over the last six years with both undergraduate and graduate students from over forty different US universities. The rest of this paper is organized as follows. Section II explores lessons learned from VLSI. Sections III and IV describe the current situation and what can be done to change it. We conclude in Section V.

II. Lessons Learned from VLSI

What made VLSI circuit design feasible to teach at the undergraduate level in the early 1980s?

1) Carver Mead's landmark textbook *Introduction to VLSI Systems* [7], which made VLSI accessible to electrical and computer engineering faculty and practitioners,
2) free computer-aided design (CAD) tools such as Magic—a circuit layout tool with a design-rule checker [6]—and simulators like SPICE (Simulation Program with Integrated Circuit Emphasis) [3], which made electrically correct integrated circuit design possible,
3) MOSIS (Metal Oxide Semiconductor Implementation Service) [1], an inexpensive (free) semiconductor fabrication service subsidized by the US government available to US universities, and
4) government-sponsored programs to teach VLSI design to engineering faculty.

The above in combination had the following effects and benefits for electrical and computer engineering faculty and students:

1) Mead's book offered simple but effective models of circuits and layouts that made teaching a one-semester VLSI course possible,
2) CAD tools such as Magic and SPICE provided an independent means of checking designs and layouts,
3) university faculty and students could make their own chips and was tangible evidence for students that they could do something recognizable by industry and government that was highly regarded and innovative, and

[1] Who actually said this is disputed, although this quote is often attributed to Benjamin Franklin or Albert Einstein.

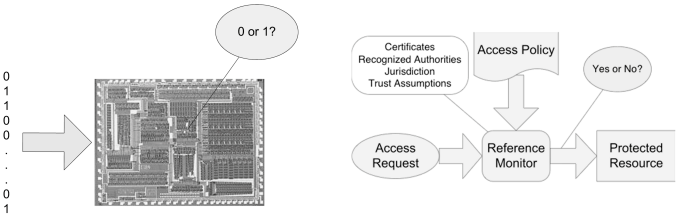

Fig. 1. Rigorous derivation of behavior

4) government-sponsored programs aimed at university faculty produced a critical mass of faculty to teach VLSI to undergraduates.

Mead's textbook was crucial in that it was the first textbook to present a continuous (and accessible) path through several levels of abstraction: (1) stick diagrams in layouts as transistors, (2) transistors as switches in circuits, (3) switches as logic and registers in finite-state machines, (4) logic and finite-state machines as data and control paths, and (5) data and control paths as algorithms.

Today, VLSI is part of the undergraduate curriculum. We can learn from the example of VLSI to move security and trustworthiness into mainstream engineering.

III. Security and Trustworthiness in Current Engineering Programs

The situation for security and trustworthiness is not unlike the state of VLSI in the late 1970s.

1) There are relatively few engineers who can do secure system design.
2) US economic and security interests are perceived to depend on the security and trustworthiness of its systems.
3) Industry is actively seeking more graduates from any field who are capable of thinking critically about security.
4) The US federal government is attempting to increase the number of students enrolling in degree programs that teach security by offering scholarships to students and recognition to US universities teaching security.

Currently, there is no consensus as to what and how security (let alone trustworthiness) should be taught at the undergraduate level. In contrast, there is considerable consensus in hardware engineering. Most electrical and computer engineering programs equip students with the following capabilities:

1) the capability to implement logic functions and storage elements as digital circuits using transistors,
2) the capability to specify, design, and verify computer hardware using propositional logic, finite-state machines, and CAD tools,
3) the capability to prototype computer hardware using discrete components, field-programmable gate arrays (FPGAs), and semi-custom and fully-custom VLSI circuits, and
4) the capability to implement algorithms in hardware.

The capabilities above are based on combining mathematical and logical representations (e.g., propositional logic and finite-state machine theory) with implementation technologies such as transistors and CMOS (complementary metal oxide semiconductor) design. This combination enables hardware engineers to do the following: when given a hardware design, the primary inputs, and the values in the registers, hardware engineers *derive* the value of signals anywhere in the integrated circuit using logic and mathematics. This is illustrated by the left side of Figure 1.

What would the equivalent situation be for security and trustworthiness? When given (1) a request to access a protected resource, (2) an access policy, and (3) assumptions about whose authority is trusted and their jurisdiction, engineers should be able to *derive* whether or not the reference monitor guarding the resource should allow access to the resource. This is illustrated by the right side of Figure 1.

Our view is calculating the *yes* or *no* access decisions is central to security and trustworthiness in much the same way as calculating the 1 or 0 digital values is central to hardware. Reference monitors protecting resources have a similar role to finite-state machine controlling the flow of data in hardware. Policies in security specify allowable behavior in much the same way as algorithms and instruction-set architectures specify how hardware should behave.

IV. Policy-Based Design

Policy and access control decisions exist at all levels of abstraction, from physical memory up through abstract information flow policies for confidentiality and integrity. Hence, engineers must be able to rigorously deal with a wide variety of access control policies and scenarios such as:

- deriving whether or not a virtual machine monitor (VMM) should grant a virtual machine's (VM) request to execute any particular instruction,
- deriving the necessary policies for a delegate to operate on behalf of someone or something, e.g., a health care proxy speaking on behalf of a patient in a coma, and
- deriving whether or not read or write access should granted to a person or process consistent with an information flow policy such as Bell-LaPadula.

Saltzer and Schroeder's oft-cited paper *The Protection of Information in Computer Systems* [9], should be the foundation of what every computer engineering student learns about secure systems design. While [9] is known primarily for the principle of least privilege, the majority of the paper is devoted to process isolation and sharing policies and mechanisms at the level of physical memory. Popek and Goldberg's paper on virtualization [8] is also part of the foundation for process isolation and sharing given the widespread use of virtualization. At the level of information flow policies, Bell and LaPadula's confidentiality policy [4] is an example of seminal work that is usually included in any description of information policies.

While the above policies and mechanisms were first developed over thirty years ago, how are they presented in ways that are amenable to formal derivation and calculation by engineers?

For several years we have been experimenting with a relatively simple multi-agent modal logic based on the work of Abadi and colleagues [2]. We have both simplified and

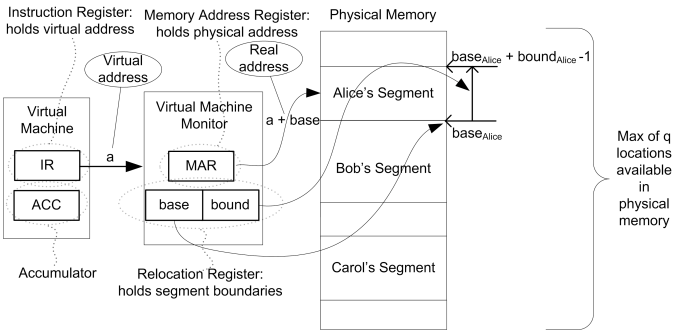

Fig. 2. Virtual Machine and Virtual Machine Monitor

extended this logic by substituting delegation for roles and adding the semantics for partially ordered confidentiality and integrity labels and levels.

Logical rules in our access-control logic have the form

$$\frac{H_1 \quad \cdots \quad H_k}{C,}$$

where $H_1 \cdots H_k$ and C are formulas in the logic. $H_1 \cdots H_k$ are the *hypotheses* or *premises* and C is the *consequence* or *conclusion*. Informally, we read logical rules as "if all the hypotheses above the line are true, then the conclusion below the line is also true." If there are no hypotheses, then the logical rule is an *axiom*.

Logical rules are used to manipulate well-formed formulas of the logic. If all the hypotheses of a rule are written down (derived) then the conclusion of the rule also can be written down (derived). If all the rules are sound (and we have proved that the rules of our logic are in fact sound), then any theorems derived using the rules are also sound.

We have successfully used this logic to teach access-control methods formally and rigorously to rising juniors and seniors from forty US universities as part of the US Air Force's Advanced Course in Engineering (ACE) Cyber Security Boot Camp [5]. Both the logic and content for an 8-hour intensive course are detailed in [5]. Since 2007, the Air Force ACE program has increased the time devoted to our formal approach to access control to 12 hours in 2008 and 21 hours in 2009.

Examples: As an example of what students in the ACE program can do, we offer three examples. At the lowest hardware level, students are given a memory access problem using a virtual machine (VM) and virtual machine monitor (VMM) as shown in Figure 2. They are asked to prove that the request to execute the instruction LDA @5 (load the accumulator with the contents of virtual address 5) should be permitted if the memory segment base is 8, the segment size is 16, and the size of physical memory is 32. The logical theorem they derive is

$$\frac{\begin{array}{c} IR \text{ says } \langle \text{LDA } @5 \rangle \qquad RR \text{ says } \langle (8,16) \rangle \\ IR \text{ says } \langle \text{LDA } @5 \rangle \supset (RR \text{ says } \langle (8,16) \rangle \supset ((8+5<32) \supset \\ ((5<16) \supset \langle \text{LDA } @5 \rangle))) \end{array}}{\langle \text{LDA } @5 \rangle.}$$

As an example of delegation, students consider a simplified health-care proxy where Bob is Alice's delegate in case Alice is in a coma. Alice's wishes are to not be resuscitated if she is in a coma. Alice designates Bob to be her delegate to say this when she cannot. She puts it in writing and signs it. Hospital policies must be aligned with Alice's if her wishes are to be honored. Also, her signature must be recognized as hers. Students are asked to (1) develop the hospital's policies, (2) state necessary trust assumptions, and (3) prove Alice's wishes will be honored.

Finally, students are given (1) Bell and LaPadula's simple security condition and *-property, and (2) a labeling and partial ordering of security levels assigned to objects and subjects, and asked to prove formally that Alice with a TOP SECRET clearance can read file *foo* at the SECRET level.

V. Conclusion

Our experience shows that undergraduates are capable of achieving the same level of rigor in designing and verifying secure systems as they achieve in designing computer hardware. The students we have taught from over forty US universities are able to reason formally about mandatory and discretionary access-control policies and decisions using the logic described in [5]. Our observations lead us to conclude that policy-based approach combined with formal logic to teaching undergraduates about security and trust is both feasible and desirable.

References

[1] Available at http://www.mosis.com/.
[2] Martin Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, September 1993.
[3] W. Banzhaf. *Computer-Aided Circuit Analysis Using PSpice*. Prentice-Hall, 1992, 2nd edition, 1992.
[4] D. Bell and L. LaPadula. Secure computer systems: Mathematical foundations. Technical Report Technical Report MTR-2547, Vol. I, MITRE Corporation, Bedford, MA, March 1973.
[5] Shiu-Kai Chin and Susan Older. A rigorous approach to teaching access control. In *Proceedings of the First Annual Conference on Education in Information Security*. ACM, 2006.
[6] Robert N. Mayo, Michael H. Arnold, Walter S. Scott, Don Starrk, and Gordon T. Hamachi. *1990 DECWRL/Livermore Magic Release*. Digital Western Research Laboratory, 100 Hamilton Avenue, Palo Alto, CA, September 1990. available at http://www.hpl.hp.com/techreports/Compaq-DEC/WRL-90-7.html.
[7] Carver Mead and Lynn Conway. *Introduction to VLSI Systems*. Addison Wesley, 1980.
[8] Gerald J. Popek and Robert P. Goldberg. Formal requirements for virtualizable third generation architectures. *Communications of the ACM*, 17(7):412–421, July 1974.
[9] Jerome Saltzer and Michael Schroeder. The Protection of Information in Computer Systems. *Proceedings IEEE*, 1975.