

SECURITY AS A COMPONENT OF A COMPUTER SCIENCE CURRICULUM AT AT LIBERAL ARTS COLLEGE

Everett L. Bull, Jr.

Pomona College

Abstract: We present the experiences of integrating topics in computer security into an undergraduate liberal arts computer science curriculum.

Key words:

Pomona College is a small, highly-selective, residential, liberal arts college located in southern California. Its 1500 undergraduate students major in traditional academic disciplines in the humanities, natural sciences, and social sciences. The computer science major prepares its students for a wide range of careers by providing them with the fundamental principles, tools, and concepts of the discipline.

We are a small program and cannot easily add new courses, although a security elective is under consideration. On the other hand, a benefit of our size is that the linkages between courses are flexible, and it is easy to add, adapt, and rearrange topics within existing courses. Customization is the norm among the faculty members.

Below, we give some examples of such customization by showing how computer security was integrated into existing courses at Pomona College in 2003–2004. Much of the material came from the Fifth Workshop on Education in Computer Security, held in June 2003 at the Naval Postgraduate School in Monterey, California

Computer security is an effective vehicle for conveying important ideas in computer science. First, security is of current interest. Students read about it and care about it. They encounter related material in courses on constitutional law, economics, and media studies. They feel the immediate effect when the network in the residence halls is the victim of an attack or when local policies affect their abilities to share files and communicate via instant messaging. Second, security cuts across the discipline of computer science. It touches on algorithms and complexity, architecture, programming languages, operating systems, and networks. Finally, security is accessible. Its basic ideas can be understood by first-year students, who can then build up their knowledge throughout their undergraduate careers.

1. CURRICULAR COMPONENTS

During the academic year 2003–2004, we introduced computer security into four places in the curriculum: our second course in computer science, a senior seminar, an intermediate course on computer systems, and a senior project.

1.1 Computer Science 52, *Fundamentals of Computer Science*

The second course in our major sequence is an attempt to give the students a broad introduction to computer science. Although it is programming-intensive, it is not *about* programming. It is about science. We emphasize recursion and correctness through the use of the functional paradigm as expressed in the language ML.

One of the first assignments was an exercise in list processing: The students were to write an “infinite precision” integer package. Later in the course, they used the package to implement RSA encryption. The students posted public keys and encrypted messages for others to decrypt. (To get things started, the author posted an encrypted passage from Dostoyevsky that contained the phrases, “I am a sick man. ... I believe my liver is diseased.” Very shortly into the assignment, the author received an electronic mail message from one of the students inquiring about his health.) Students learned first-hand the relationship between complexity theory and cryptography. In particular, those who did not follow instructions and implemented exponentiation as iterated multiplication learned a lot about the difference between linear and exponential time.

In the section on data representation, we briefly discussed graphics and saw a demonstration of steganography. A brief presentation had been developed by the author for another purpose, a presentation to the local senior citizens computer club. There were a series of images in which short stories by Edgar Allen Poe were hidden in a photograph of the author’s dog. Even though the students knew what to expect, they were satisfied to see it so clearly. With Poe occupying only one bit per byte, the image of the dog was indistinguishable from the original. At three bits per byte, it was only slightly fuzzy. At six bits, it was Warhol-esque. The surprise was that the dog was still distinguishable even with seven bits of each byte allocated to Poe.

One of the later parts of the course was devoted to an introduction of assembly language and computer organization. We studied the stack discipline for subprogram calls and talked about buffer-overflow attacks. Although we did not go into the mechanics in great detail, we did discuss how strongly-typed languages like Java and ML can reduce the possibility of stack smashing.

We spent about seven one-hour classes on security-related material.

1.2 Computer Science 190, *Senior Seminar*

The fall senior seminar is required of all computer science majors. Faculty members select two topics and collect readings on them. Students read the papers and take turns summarizing and leading discussions.

In the fall of 2003, we chose the topics of Evolutionary Computing and Computer Security. The students found the security readings challenging. They acquired a firmer grasp of basic principles and gained a new vocabulary with which to discuss security issues.

A list of the computer security readings appears in the appendix. We spent a total of four two-hour meetings on security in the fall semester.

1.3 Computer science 105, *Computer Systems*

Our course uses the recent text *Computer Systems: A Programmer's Perspective*, by Bryant and O'Hallaron (Prentice Hall, 2003). It uses the question "How does a program execute?" to expose architecture and instruction sets, assembly language, compilers, operating systems, and network concepts. Two exercises, due to Bryant and O'Hallaron and not the present author, were closely related to security.

One assignment was an exercise in reverse engineering. The students were given an executable program called a "bomb" and used debuggers and decompilers to find the input strings which would prevent the program from "exploding." The other assignment had a similar theme, but this time the students launched successful buffer attacks to prevent explosions.

We spent approximately five seventy-five minute class meetings on this material in the spring semester.

1.4 Computer Science 191, *Senior Project*

Each graduating senior is required to carry out a project of his or her own design. One senior, Adam Carasso, was interested in file systems, specifically the secure deletion of data. His initial idea was to achieve *complete* erasure of files, but preliminary research convinced him that it was impossible and that further work would require a magnetic force microscope, so he shifted to encrypted file systems. His goal was to design and implement file system for Linux based on public key encryption. There was not sufficient time for implementation, but he did complete a design of a hybrid system.

Not surprisingly, key distribution and management was a central concern. Efficiency of public key systems was also an issue. To mitigate the performance penalty, Adam chose to encrypt data with a symmetric key algorithm, using the public key only to encrypt the symmetric key. The encrypted symmetric key was stored in a special data structure in the file system. He was therefore able to preserve random access to files. Also, vulnerability was reduced because only symmetric keys are written to memory, and the loss of one such key would reveal only a small portion of the data. The private key, whose secrecy is more critical, is stored on a smart card and never written to memory.

Adam worked independently throughout the spring semester; neither he nor anyone else knows how many hours he spent.

2. OTHER OPPORTUNITIES

In the courses just described, we took advantage of opportunities of the moment. In another place or another time, we would see a different set of possibilities. For example, there is a place (one could easily argue it is an *essential* place) for formal systems in our sophomore-level course entitled *Computation and Logic* or in *Large-scale Software Development*. Similarly, one might expect to see a discussion of language safety and type safety in *Programming Languages*. In the *Computer Systems* course mentioned earlier, it would be natural to include material on network security.

The point is not to make security the theme of every course in the major, nor is it to replace a focused, in-depth course on computer security. Rather, every undergraduate computer science major should that questions about security and privacy are integral to the discipline and can be found in nearly every subfield. Our examples are meant to illustrate that there are many "teachable moments" which an instructor may leverage.

Like computer security, material on ethics and professionalism may be integrated into the mainstream courses. The two belong together. It would be wrong to teach techniques of reverse

engineering without also mentioning intellectual property, to teach network snooping without also discussing privacy, or to teach stack smashing without also delineating social and legal responsibilities. The curriculum guidelines of professional organizations all recommend that there be consciously-designed components of society, morals, and law. We believe that some, but perhaps not all, of that material ought to be woven into central courses and not relegated to separate and easily-ignored modules.

3. CONCLUSIONS

Our experience shows that it is possible to introduce topics in computer security into standard, existing courses. Computer security can motivate and illustrate the primary topics in a course, just as the central subject matter can illuminate security. The cost, in terms of class time, is minimal because security examples are often direct substitutes for other kinds of illustrations. Our approach is not a substitute for an advanced security elective in the undergraduate curriculum; it is instead a way to reach a larger group of students—not just those who choose a specialized course.

Any introduction to computer security at the undergraduate level should be grounded in the *principles* of computer science. Current events and excitement about hacking techniques can provide energy and motivation, but they should not overwhelm a class. For the sake of both computer science and computer security, we want our students to come away with a lasting understanding of the fundamental concepts of the discipline.

APPENDIX. SENIOR SEMINAR READING LIST

The papers chosen for the senior seminar are not necessarily the “best” or “most influential.” We select papers that represent a wide variety of topics, that come from different periods in the development of computer science, and that are written for various purposes. The final list includes some true classics, a few popular articles, and several narrowly focused research papers. Part of the exercise is to develop the student’s ability to read critically and to discriminate among the papers. Accordingly, our list should not be taken as a recommended or comprehensive set of readings.

For the security section in the fall of 2003, the seminar participants read and discussed the following papers:

- Thompson, “Reflections on Trusting Trust,” *Communications of the ACM*, Volume 27 (8), August 1984, pp. 761–763.
- Lampson, *Computer Security in the Real World*, Marshall D. Abrams Invited Essay, presented at the Annual Computer Security Applications Conference, 2000. A revised version with the same title appears in *Computer*, Volume 37 (6), June 2004, 37–46.
- Ellison and Schneier, “Ten Risks of PKI: What You’re not Being Told about Public Key Infrastructure,” *Computer Security Journal*, Volume 16 (1), 2000.
- Anderson, “Why Cryptosystems Fail,” *Proceedings of the First Conference on Computer and Communication Security*, ACM, 1993.
- Borisov, Goldberg, and Wagner, “Intercepting Mobile Communications: The Insecurity of 802.11,” *Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking*, ACM, 2001.
- Lampson, “A Note on the Confinement Problem,” *Communications of the ACM*, Volume 16 (10), October 1973, pp. 613–615.
- Gong, Mueller, Prafullchandra, and Schemers, “Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2,” *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, Monterey, California, December 1997.
- Lampson, “Protection,” *Proceedings of the Fifth Princeton Conference on Information Sciences and Systems*, Princeton, 1971. Reprinted in *ACM Operating Systems Review*, Volume 8 (1), January 1974.
- Schneider, Morrisett, and Harper, “A Language-Based Approach to Security,” in *Informatics—10 Years Back, 10 Years Ahead*, Lecture Notes in Computer Science, Volume 2000, Springer-Verlag.

These papers were included in the readings, but there was not time for discussion:

Lamport, "Password Authentication with Insecure Communication," *Communications of the ACM*, Volume 24 (11), November 1981, pp. 770–772.

Bellovin, "Security Problems in the TCP/IP Protocol Suite," *Computer Communication Review*, Volume 19 (2), April 1989, pp. 32–48.

Schneider, "Open Source in Security: Visiting the Bizarre," *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, IEEE Computer Society, May 2000, pp. 126–127.