Teaching Computer Security

Matt Bishop

Department of Mathematics and Computer Science, Dartmouth College, 6188 Bradley Hall, Hanover, NH 03755-3551

Abstract

This paper describes three courses in computer security, and offers come comments about their appropriateness, their advantages and their disadvantages.

1. INTRODUCTION

Interest in computer security has grown in the past few years [14], and nowhere is this more evident than in colleges and universities. Perhaps a rising awareness of potential threats encourages students to explore this area; perhaps the desire of businesses and industry in general to improve the security of their systems leads students to hope their studies will help them get jobs; or perhaps students enjoy the game of defeating security mechanisms and then improving them to keep others out. In any case, computer security has become a "hot" topic.

While security is discussed in some classes (such as operating systems courses), it has so many facets that only a course devoted to computer security can examine the basic underlying principles in detail. For example, the notion of "auditing" involves both operating system security and database security, yet rarely is this more general notion discussed in database or operating systems classes. This is hardly surprising given the number of subareas of computer science (programming languages, architecture, operating systems, software methodology and engineering, database and information retrieval, and human-computer interaction [5]) in which security considerations may be involved.

A course on computer security unifies the threads running through these subareas. By bringing together seemingly unrelated (or distantly related) parts of these subareas, students learn how integral computer security is to computer science, and how deeply it involves mathematics and computer system science. The depth to which these relationships are explored depends, of course, on the time available and the level and quality of the students. In addition, relationships to fields of study other than computer science, such as ethics, law, and public policy, often interest students as much as (if not more than) the technical aspects of the class.

As we use the term "computer security," we mean both mathematical cryptography and other techniques for protecting a computer system (including protocols which use cryptography); the latter we shall refer to as "computer systems security." Although often taught in the same course, these two topics are rich enough that they could easily be taught separately, as [10] suggests.

A paradigm for teaching computer security is presented in [10], in which three courses are

From the *Proceedings of the Ninth IFIP International Symposium on Computer Security, IFIP/Sec '93*, pp. 43-52 (May 1993).

recommended: a survey course, followed by advanced courses on cryptography and system security. Several papers describing such courses [1,11,15,16] present descriptions of the material covered; some (notably [1]) presents a rationale for the topics, and others [1,6] justify the need for such courses. In this paper, we take this need for granted, and focus instead on what we have taught and why. The courses described here fall under the paradigm in [10], but this is fortuitous. Further, we found each course has its benefits and drawbacks, and shall discuss these in the conclusion.

2. The Survey Course

This course (taught in 1989) covered both cryptography and computer system security, and both graduate and undergraduate students attended. We used [4] as the text and supplemented it with a variety of papers.

Because students would have to analyze various protocols and algorithms, and use elementary number theory, the prerequisites included a course in analysis of algorithms (which, in turn, required a working knowledge of a higher-level programming language). We did not require an operating systems course, but did expect all students to have completed a course in computer architecture, and when needed spent a few minutes discussing how the relevant part of the operating system worked. This seemed to be quite sufficient. Finally, all students had to have taken a course in software engineering, which proved invaluable when we discussed formal verification and its practical implications (as well as the complexity of integrating security mechanisms into systems).

We followed the general outline of the text, although lectures brought the material up to date. Mathematical cryptography and cryptographic protocols occupied the first month, and computer system security (except cryptographic protocols) occupied the second. In this part, we addressed disclosure and integrity using access control, and its compromise using malicious logic and covert channels, and analysis of a system to determine if confidentiality could be compromised. From this we discussed information flow models on which this analysis was based, and techniques for performing such analyses, which led to program verification. We concluded with techniques for compromising statistical databases.

As we firmly believe students learn best when having fun, we emphasized how to break ciphers in the first part of the class ([7] supplied many examples), and drew justification for countermeasures and protocols from history ([12] was outstanding in its history of codes and codebreaking). The system security part drew examples from well-known security violations such as the Internet worm (which introduced a discussion of malicious logic). As computer viruses had begun appearing on campus, we spent a day talking about how the various types of viruses worked, and what countermeasures were used, as well as other techniques being explored but not yet ready. The class found the question of what a computer virus on a multiuser machine could do fascinating, especially since those machines had protection mechanisms already in place.

At Dartmouth, we are fortunate to have a computing center with a well-known and well-established code of ethics. When we began discussing system security, we first looked at the Computing Code of Ethics [18] and discussed what would and would not be allowed. Throughout the class, we would point out questions of ethics, not to resolve them, but merely to make students aware of them. No official or computer center staff member ever had even to repri-

mand a student in this class; whether that speaks well for discussing ethics during the class, or the morals of the students, or both, is left for the reader to ponder.

Each student was required to present research into a topic of their choice. Here the liberal arts orientation of the College was clear; one of the presentations discussed the relationship of laws to computer security violations, and another was a talk on a Soviet cipher used in the 1950s (this was presented by a student majoring in both computer science and Russian). This work deepened students' appreciation of how computer security affects, and is affected by, its environment and societal and legal concerns.

Because there were both graduate and undergraduate students taking the course, the survey format seemed to work well; the students understood the more formal mathematics (notably in the theory and cryptography sections), and all the students liked the course overall.

3. The Cryptography Course

Because mathematics and computer science are taught by the same department at Dartmouth, we decided to teach a course on mathematical cryptography. This was in part a response to those students who enjoyed the survey course on computer security, and in part because other students had been asking about the mathematical foundations of cryptography. The author and a colleague (J. Shallit) collaborated on the course, dividing it into two parts.

The first part, which the author taught, covered information theory and classical cryptography. Because information theory is central to understanding the limits of cryptography, and classical cryptosystems are so widely used and combine both pure and applied mathematics and statistics, we believed it essential that the course cover these topics as well as the more modern methods of cryptography (which made up the second part of the course, taught by our colleague).

The classical portion covered basic information theory, monoalphabetic and polyalphabetic substitution, linear feedback shift registers, rotor systems, and the Data Encryption Standard; students are taught both how the ciphers work and how to cryptanalyze them. The course text for this portion [13] provided the framework and many of the analytical techniques for this part of the course, although we supplemented it with outside reading as appropriate (for example, the students read several papers from recent conferences when we discussed the DES). Our goal was to show first, the building blocks upon which classical ciphers are based; second, how ciphers evolved historically; and third, how classical ciphers could be attacked. This part of the course included discussions of famous compromises in history, and anecdotes illustrated the thesis that cryptosystems which may seem unbreakable usually are not.

All the students seemed to enjoy breaking ciphers. The techniques discussed in class ranged from the statistical to the algebraic; further, to encourage students to play with simple cryptanalysis, and to add spice to the course, every day students were given a cryptogram, and invited to break it. These ranged from the very simple (a portion of the poem "The Walrus and the Carpenter" enciphered using a simple substitution cipher) to the challenging (a transposition, then a substitution). If the text being enciphered was not well known, an introduction described the topic of the plaintext; to keep the puzzles fun, hints were given on the back.The students enjoyed these; most got the early ones, and a few deciphered all of them. These "Quicktograms" were also chosen to provide a starting point for the day's discussion (for example, the transposition/substitution cipher led very naturally to a discussion of product ci-

phers, of which the DES is one). They also provided a very brief, valuable review of material covered in the first day or so.

One drawback of the classical portion was that most of the students had trouble with the statistical techniques used to cryptanalyze ciphers. This suggests that when the course is next taught, the instructors should review basic statistics, Bayesian decision functions, generating functions, and probability theory. The alternative, requiring these as prerequisites, is unnecessary as the material can be covered fairly quickly when discussing their use; for example, the nature of correlation can be explained when using it to break a Cæsar cipher.

The second part of the course, modern cryptography, began with public-key cryptosystems and a review of number theory. Then we discussed pseudorandom number generation, probabilistic encryption, applications, zero-knowledge proofs, and quantum cryptography. Numerous papers augmented the text [3]. In this portion of the course, the "Quicktograms" were discontinued, as little cipher-breaking was done due to the complexity of the cryptosystems (but potential weaknesses were thoroughly discussed; again, the students seemed to enjoy this most of all).

Each student also had to present a paper on some aspect of cryptography. Again, the students seemed to learn quite a bit from this, and some of the talks were excellent. The papers presented ranged from using cryptography in an electronic banking network [9] to randomized encryption techniques [17], and each presented a facet of cryptography or its applications not covered in class.

Because of the mathematical subject matter, students had to have a course in abstract algebra. Many modern cryptosystems are quite breakable theoretically, and draw their strength from the computational infeasibility of the attack. Since the course discussed why the attacks were often computationally infeasible, students also had to have a course in the analysis of algorithms. As discussed above, had students had a course in statistics and/or probability, they would have found the course easier; but this should probably have been strongly recommended rather than a prerequisite.

Again, student reaction to the course was good. We were able to explore the subject in depth, but there is an abundance of material that we did not cover; this could easily have been two separate courses, one on classical cryptography and the other on modern cryptography.

4. The System Security Course

This covers very little cryptography, but focuses on other aspects of computer security. It is somewhat similar to the one described in [1], but differs in emphasis and topics; we discuss the mathematical foundations of computer security somewhat more than the course in [1] appears to, and cover a somewhat different list of topics. We should point out that this course was first taught as a seminar course in 1990; it will be taught as a lecture course in January 1993, so what is presented is the outline of the course from 1990 as modified for 1993.

We started with a discussion of the problem: what is system security and why are computers not secure? This led naturally to a discussion of policy and mechanism, and general design principles. During this time we discussed "crackers," privacy, responsibilities of users, programmers, and system managers, the ACM Code of Ethics [19], and the Computing Code of Ethics. In this way, students were given guidelines on what was acceptable and what was not, and whom to talk to should they need help to resolve a problem.

Computer system security consists of three parts: the prevention of illicit disclosure (*confidentiality*), the prevention of illicit altering (*integrity*), and the availability of resources *(non-denial of service*). Central to all these is the concept of ownership. As the identity of the owner of and object on a computer system controls access to that object, the discussion on privacy and ethics led directly into how users and systems could identify each other; we answered by comparing different authentication schemes using the framework in [2]. Examples were drawn from systems the students used daily as well as other sources; and students were often asked to analyze these mechanisms for weaknesses, and then either suggest remedies or figure out a better design.

Then came formal modelling, the goal being to see how easy or hard it is to provide security in a theoretic sense. We discussed the Harrison-Ruzzo-Ullman result, and access control models. Using these, we examined the different types of access control (mandatory vs. discretionary) and access control mechanisms (capabilities, access control lists, *etc*.) as well as how these were implemented in computer systems.

Because access controls limit both disclosure and modification, we sprang from security to integrity. Here, we again began with a survey of formal models. Implementing these models requires a notion of "trust" or "certification" of programs as being of high integrity (see [8], pp. 70-71, for a marvelous discussion of the relationship between trust and integrity); to demonstrate the threat, we discussed malicious logic (Trojan horses, computer viruses, and their ilk) and showed how each model would deal with such programs. We then showed how important the assumptions were by showing how the system could be subverted if trust were misplaced (as an introduction to this issue, we used an incident in which an infected anti-virus program was used to disinfect a personal computer).

The trust required is twofold; in addition to trusting the certification process (or the software on the system), one also needs to trust the system security mechanisms to implement the model correctly. We began with the second part, looking at the notion of reference monitors and security kernels, and how trusted path techniques worked. We then tackled the first part, looking at the formal specification and verification of systems, especially at methodologies and tools for proving specifications, how specifications could be implemented accurately, and compiler and information flow techniques for analyzing potential paths of disclosure.

Because most computer systems are connected to other systems, we next looked at issues in network security. We began by examining what security services should be offered and when they could be used, again drawing examples from various Internet protocols and international protocol suites.as well as Kerberos. We also looked at security in distributed computer systems and file systems, again using existing implementations as examples.

As in the survey course, ethical, legal, and social issues were not simply discussed at the beginning, but also throughout the class. Again, no member of the class caused any security problems.

Both graduates and undergraduates took the 1989 seminar course; they were able to select papers which they found interesting, and (fortunately) their interests were diverse enough so each student was able to present 5 or 6 papers during the term. In addition, we presented a number of lectures about each topic, to provide background before any papers were presented and to provide additional information when the papers chosen did not represent the topic adequately.

Interest in the 1993 version of this course, which will be a lecture course, appears to be high. In addition to homework and examinations, each student will be required to do a project.

It could be a term paper or (preferably) an experiment. If the latter, and if there is any possibility of conflict with the Computing Code of Ethics, the student or students will need to obtain the approval of the appropriate staff members before beginning. In this way, they will learn what precautions must be taken to preserve the privacy of users not directly involved with the experiment.

5. Discussion

Computer security consists of three subtopics: techniques to ensure confidentiality, techniques to ensure integrity, and techniques to prevent denial of service. The survey course touched on all three, though not in too much depth; the cryptography course focused on confidentiality, with a smattering of integrity; and the system security course went into greater depth in all three subtopics. The system security course used cryptography as a "black box," discussing protocols but assuming that students understood how cryptosystems worked (a short review was given at the beginning of the term). This emphasized that cryptography is not only mathematically very rich, but that it also is a valuable tool for protecting confidentiality and integrity. But it must be supported by an appropriate system structure, which was the focus of the system security course.

The emphasis in the content of the three courses was also quite different. The survey course did not place much emphasis on mathematical formalism; while mathematical proofs were given as needed (especially in the cryptography part), informal explanations predominated. Further, the course was somewhat oriented around computer applications. The mathematical cryptography course looked at mathematical foundations, with much formalism and little emphasis being placed upon computer applications (none of the exercises required a computer); the mathematics students were quite comfortable with it. Similarly, the system security course covered the mathematical description of several types of systems, and then explored how the mathematics was translated into programs (or hardware). A fair amount of mathematical sophistication was required here as well.

Each course was aimed at a different group of students. The survey course was taught at a senior undergraduate/first-year graduate level, and the two juniors who took it had no problems with the work. We intended the mathematical cryptography course to be a first-year graduate level course, but several graduate students found the course quite difficult (however, one senior undergraduate who took it found the course less difficult and scored higher than the graduates). By contrast, the system security seminar students ranged from a junior to two second-year graduates, the common characteristic being they were all interested in some facet of computing systems. The courses were not seen as a sequence; only one person took two, and no-one took all three.

Which course is most useful? If by "useful" we mean the one that will teach students that amount of computer security appropriate for computer scientists not specializing in computer security, the survey course is most useful; one can analogize to the difference between a one-term survey course in computer graphics and a three-term sequence in which the full mathematical and architectural underpinnings of that subject are explored. If by "useful" we mean the one which is most likely to be applied to a student's current research, then taking either of the two advanced courses after the survey one is best. For students whose primary interest is in either history or mathematics, the mathematical cryptography course was the more appropriate

advanced course as it demonstrated techniques and analyses drawing on many fields (algebra, statistics, *etc*.). For students of computer science, especially computer systems, the course on system security was more appropriate, because it discusses approaches used by system architects.

6. Conclusion

This paper has presented brief descriptions of several courses in computer security. We have described the contents and prerequisites of each, and added our personal experiences in teaching them. The courses have been successful, being well attended and well received. The specific nature of the courses taught grew from our own interests and from the requests of the students.

The division of computer security into confidentiality, integrity, and denial of service subtopics suggests an alternate approach to organizing the content of computer security courses. Thus, there would still be a general survey course, but the advanced courses would be on techniques for confidentiality and techniques for integrity. The advantages to such an arrangement would be that the courses would have a central theme and could explore the application of technique in different environments (for example, cryptography as a tool for confidentiality and a tool for integrity). The problem, of course, is that many of the techniques use the same mechanism. Would access control be discussed in detail in the integrity or confidentiality course? If all three courses were carefully planned, and students were expected to take them as a sequence (or had to be willing to do extra work), this arrangement would work well. Without careful planning, it would not. But there is something quite appealing about eliminating the distinction between cryptography as one component of computer security, and everything else as another.

What lies ahead? It is unlikely that a formal three-course sequence in computer security will become a reality at Dartmouth College, simply because of staffing requirements. However, the author does plan to continue teaching them periodically, scheduling allowing. We hope these courses will show students the foundations of computer security, how security can be integrated into computer systems and, perhaps most important, make students more aware of the need for security in the design of computer systems.

7. References

1 A. Arsenault and G. White, "Teaching Computer Systems Security in an Undergraduate Computer Science Curriculum," *Proceedings of the Fourteenth National Computer Security Conference* (Oct. 1991) pp. 582-597.

2 M. Bishop, "Password Management," *COMPCON 91* (Spring 1991) pp. 167-169.

3 G. Brassard, *Modern Cryptology: A Tutorial* (Lecture Notes in Computer Science #325), Springer-Verlag, New York, NY (1988).

4 D. Denning, *Cryptography and Data Security*, Addison-Wesley Publishing Co., Reading, MA (1984).

5 P. Denning, D. Comer, D. Gries, M. Mulder, A. Tucker, A. Turner, and P. Young, "Computing as a Discipline," *Communications of the ACM* **32**(1) (Jan. 1989) pp. 9-23.

6 K. Forcht, "The Need for Including Data Security Topics in the College Business Curriculum," *Security and Audit Control Review* **4**(3) (Summer 1986) pp.9-11.

7 H. Gaines, *Cryptanalysis: a Study of Ciphers and their Solutions*, Dover Publications, Inc., New York, NY (1956).
8 M. Gasser, *Building a Secure Computer System*, Van Nostrand Reinhold Co, New York, NY (1988).
9 D. Gifford and D. Spector, "Case Study: An Electronic Banking Network," *Communications of the ACM* **28**(8) (Aug. 1985) pp. 797-807.
10 J. Higgins, "Information Security as a Topic in Undergraduate Education of Computer Scientists," *Proceedings of the Twelfth National Computer Security Conference* (Oct. 1989) pp. 553-557.
11 H. Highland, "A College Course in Cryptography and Computer Security," *Security and Audit Control Review* **1**(2) (Spring 1982) pp.34-37.
12 D. Kahn, *The Codebreakers*, Macmillan Publishing Co., New York, NY (1967).
13 A. Konheim, *Cryptography: A Primer*, John Wiley & Sons, New York, NY (1981).
14 National Research Council, *Computers at Risk: Safe Computing in the Information Age*, National Academy Press, Washington, DC (1991)
15 B. Neugent, "A University Course in Computer Security," *Security and Audit Control Review* **1**(2) (Spring 1982) pp.17-33.
16 T. Richards, "A Seminar in Computer Audit and Control Systems," *Security and Audit Control Review* **3**(1) (Spring-Summer 1984) pp.6-10.
17 R. Rivest and A. Sherman, "Randomized Encryption Techniques," *Proceedings of Crypto '82* (1982) pp. 145-163.
18 B. Strohbehn, ed., *Student Handbook*, Dartmouth College, Hanover, NH 03755 (1992).
19 E. Weiss, "Self-Assessment Procedure XXII," *Communications of the ACM* **33**(11) (Nov. 1990) pp. 110-132.

8. Appendix. Outlines of the Three Courses

This section contains outlines of the three courses described above. At Dartmouth College, the academic year is divided into four terms of 9 or 10 weeks each, with classes either being 65 minutes three times a week or 100 minutes twice a week.

Survey Course

week 1. history of computer security and cryptography, information theory, number theory
week 2. transposition, monoalphabetic and polyalphabetic substitution ciphers, product ciphers
week 3. Data Encryption Standard, exponentiation ciphers, public key ciphers, block ciphers
week 4. encryption on a network, key management, threshhold schemes
week 5. access control models, mechanisms; malicious logic, denial of service, covert channels
week 6. verifiably secure systems, theory of safe systems, models
week 7. information flow models, flow control mechanisms, program verification
week 8. statistical databases: inference, trackers, other attacks
week 9. blocking attacks on statistical databases; class presentations

Mathematical Cryptography

week 1. history and foundations of cryptography, information theory, monoalphabetic substitution
week 2. monoalphabetic and polyalphabetic substitution
week 3. linear feedback shift register systems, rotor systems
week 4. the Data Encryption Standard
week 5. introduction to public key cryptography, necessary number theory
week 6. RSA, Rabin's scheme, Shamir's secret sharing, pseudo-random number generation
week 7. probabilistic encryption, applications (authentication, digital signatures, identification schemes)
week 8. coin flipping, zero-knowledge proofs
week 9. quantum cryptography; class presentations

Computer System Security

week 1. risk analysis, definition of security, general concepts and design principles, relevant standards (American government, commercial, and international)
week 2. basic cryptography and applications; authentication techniques
week 3. formal security models (Harrison-Ruzzo-Ullman, Take-Grant, ESPM, *etc*.) and their interpretation and application to systems
week 4. access control and multilevel security; covert channels
week 5. integrity models and their interpretation and application to systems; malicious logic
week 6. security kernels, reference monitors, trusted path, implementation issues
week 7. formal specification and verification of systems
week 8. network security, Internet and international standards and protocols, network architectures and security services, distributed system security
week 9. database security, inference; tracker attacks, countermeasures, the cell suppression problem and solution