Incorporating Security Issues Throughout the Computer Science Curriculum

Gregory B. White, Willis Marti, Mark L. Huson
Department of Computer Science, USAF Academy, CO, USA. E-mail: {white,huson}@cs.usafa.af.mil. Texas A&M University, USA. E-mail: willis@cs.tamu.edu

Key words: Teaching, Computer Security, Security Education, Undergraduate

Abstract: The need for computer professionals with an understanding of security is becoming increasingly evident. Where once news about security flaws could only be found in professional publications, today such holes are likely to become front-page news. With an increasingly computer literate population, interest in security is growing. This is especially true as companies regularly transact business on the Internet. Many schools provide Internet access so students can conduct research, obtaining information more current than was ever possible before. People regularly access the Internet to send email, participate in chat sessions, and to stay informed on current events. With such a reliance on the Internet, problems with its security are a major concern to millions. While the general public is becoming more aware of security issues, what are our universities doing to produce graduates ready to address our security needs? Computer science as a discipline has matured to the point that students are regularly in tructed in software engineering principles--they learn the importance of life cycle issues in the development and maintenance of software. Where are they receiving similar instruction on security concerns in the software life cycle? The authors propose that security should be taught throughout every computer science curriculum--that security should always be a concern and should be considered in the development of all software just as structured programming and documentation are. Based on their experience in academia, the authors present ways that security can be included in many common computer science courses. This is done not at the expense of other important topics but rather in a manner so that security is often used to illustrate and even teach these other concepts.

1. INTRODUCTION

It may be argued that the purpose of computing systems is the delivery of services to users. Computing systems consist of applications on servers connected by a network to devices which provide an interface to the users. The argument can be extended to state that the purpose of computer science, especially at colleges and universities, is to discover and teach how to create the computing systems to provide the services required by the users. As such, computer science education generally concentrates on teaching that which will make their graduates successful in this environment. Courses in computer science focus on particular aspects of creating computing systems within this overall task. Each course covers an area in detail while also discussing how the area relates to others as well. Topics such as networks, operating systems, software engineering, and databases are thus mainstays in computer science departments throughout the country. With the increase in concern about the security of our computer systems and networks, computer science graduates should also receive exposure to discussions about security.

The question arises as to what is the best way to introduce security into the curriculum. Adding another required course to already full computer science programs is not generally considered a viable solution. This is not to say that a separate course in security or cryptography should not be offered for those wishing to specialize in this important area of computer science, but rather that recommending that such a course be required will not be well received at most universities. Instead, what can be done is to introduce security at the appropriate point throughout all courses. Not only does this insure that students will be exposed to security topics, it has the added benefit of stressing the importance of security through simple repetition. All too often students will forget what they learn in a course once the final has been taken and projects turned in. By including it throughout the curriculum we inure that the topic is not soon forgotten.

It should be noted that the authors are not recommending a haphazard approach to teaching security – i.e. they are not suggesting that individual instructors add security to their courses only if they feel like it. While there is always value in mentioning security, doing so in this manner would not accomplish the real goal. Instead, what is recommended is for departments to meet to decide where best to cover the necessary topics. This, of course, leads to the question of what is necessary. At a minimum, departments should be teaching something about authentication (including common problems with passwords) and authorization, the basics of cryptography, basic network security hazards, and information about virus infection and protection. An argument can also be made to include security requirements in all software specifications.

2. INTEGRATION INTO SPECIFIC COURSES

Before determining where topics might be placed in an undergraduate curriculum, it should first be decided what concepts (relating to security) students should have an understanding of by the time they complete their undergraduate program. The authors recommend the following as the basic security knowledge all computer science majors should be exposed to in an undergraduate curriculum:

– Fundamental computer security principles (confidentiality, integrity, availability)
– Risk analysis (an understanding of basic trade-offs)
– Authentication
– Access controls
– Basic principles of cryptography
– Knowledge of the types of malicious software that exist
– Basic network security (including a discussion of web security)

These topics will not all be addressed in a single class but spread among the many courses required in the student's course of study. The fundamental security principles along with risk analysis should be addressed early in the program so that they can be referred to throughout the program. Malicious software, because of its widespread exposure in the media, is another candidate to address early in the curriculum but should also be addressed later in greater detail when the students have more programming experience. There are natural points to address some of these topics in the curriculum, such as access controls in an operating systems course or network security in a networks or data communication course. Some topics may be addressed when the opportunity presents itself. Consider, for example, briefly discussing authentication when passwords are handed out for access to a campus-wide or department network.

2.1 Introductory or Core Computer Science Courses

Many majors today require an introduction to computers or programming. Since computers are found and used in every discipline, an introduction to computer programming is no longer considered the sole domain of computer science departments. (Please note that while reference is made throughout this paper to computer science departments, the ideas apply equally well to computer engineering, computer technology, and management information systems programs.) Though it is not possible to go into great detail about security in an introductory programming course, introduction to computer science and programming courses are excellent places to start planting the seed that security should always be a consideration. There are many topics which should be mentioned that will also often generate much interest and stimulate classroom discussions. Topics such as the cracking of passwords and what constitutes a good password can raise awareness of this common vulnerability. Many students at this level are interested in privacy and discussions about cryptography can also easily fit into introductory courses. In introductory programming courses a possible assignment might be to have the students create a simple encryption program (a simple Caesar cipher such as the widely available rot13.c program which shifts all letters 13 spaces) which they can then use to encrypt files. This will both illustrate how encryption works as well as giving the students practice with simple file I/O and iteration. Introductory courses are also an excellent place to introduce discussions about ethics. Depending on the platform used in the class, discussions about file protections and permissions or email can be used to illustrate points that an instructor might want to make regarding whether it is 'right' to browse through somebody's mail and files.

2.2 Courses on programming languages

Many disciplines require their majors to take a programming language course, such as C or C++, after a freshmen introductory core course. This is also common in computer science programs as an introductory course in the discipline. These courses provide ample opportunities to introduce security topics. Instructors in these courses are always looking for interesting problems that will capture the interest of the students while still illustrating certain principles. Some examples of security related programs include a more extensive cryptography program to encrypt and decrypt files than was developed in the introductory course and a code virus which would infect source code programs. The virus program serves to illustrate how viruses can propagate, how they infect other programs, and how anti-virus software functions to detect viruses. There is no real fear that a source-code virus will do any real damage since they are easily detected, are easy to find, and require infected programs to be compiled before they can function (all reasons why actual viruses don't operate at this level). Another interesting idea for programming courses is to have students write programs and deliberately violate guidelines when they test them (buffer overflow, out-of-range values, invalid input) to see how their programs will react. In C, for example, buffer overflows have caused many security problems and showing students at this level the dangers of them will help reinforce software engineering principles which call for error checking. If students are taught early on about the dangers of not performing error checking then we may ultimately be able to reduce the number of problems caused by such errors.

2.3 Computer Architecture and Assembly-level Programming

In some departments, assembly level programming is taught (or at least discussed) in architecture courses. Other departments may actually have or require a separate assembly language class. Either way, a course in which assembly level programming is taught is an excellent place to have the writing of a virus as a project. This virus, as opposed to the high-level language virus discussed earlier, could actually be dangerous and extreme care must be taken. Some instructors may feel that the risk is too high and decide to not include such a lab in their course—this is definitely the safer approach but understanding how a virus attaches itself and where it is placed can be used to discuss and illustrate the compile, link, and load process. Other possible security related architecture projects might include researching ‘bugs’ in current chips (such as the famous problems with the Pentium chip) or using/discussing the advantages of parallel processors or developing specialized chips (such as was done to crack the Data Encryption Standard recently) in the cracking of passwords.

2.4 Algorithms

Encryption algorithms can again play a role in teaching both core computer science concepts and security in a traditional course on algorithms. Implementing and analyzing specific encryption algorithms provides an easy mechanism in this regard. Comparison of the various types of encryption algorithms and the methods that would be needed to break them provides ample opportunity to discuss many points.

2.5 Database

Databases are one of the most common and important applications in computer science. Since the goal of many intruders is to obtain as much information as possible about the target, its employees and its customers, databases become a particularly attractive target. Consequently it is extremely important that students in a database class be exposed to the security issues relating to them. Issues such as data validation, information integrity, and privacy can easily be inserted into many discussions. A lab illustrating the problem with users being able to infer ‘classified’ or protected information by observing other obtainable pieces of data from the database can be used to illustrate the challenges inherent in database security.

2.6 Operating Systems

One of the most natural and applicable classes to address security issues is in an operating systems course. Since one of the tasks assigned to most operating systems is security, a course on this topic would not be complete without a discussion of it. Possible security projects and labs abound in this area. These include the design of a multi-level secure operating system in which the students are asked to consider and discuss the implications of this environment on the file system, authentication, authorization, memory, and context switching. Discussions or implementations of portions of a remote file system where issues such as authentication/belief need to be addressed and are other possible projects. If a lab environment exists to support it, an assignment in which students 'hack' a password file (such as is possible in LINUX) using boot disks would allow an instructor to demonstrate that often if physical access to a machine can be obtained, then security is hard to maintain. This same assignment can also be used to demonstrate basic principles of operating system design and abstraction since only certain commands (resident in the operating system kernel) would be usable and many commands/programs which students might normally use would not work since they reside at a 'higher' level.

2.7 Networks and Data Communication

Equally as important as the operating system course, a course on networks and data communication would not be complete without discussions of security. This is also a course in which tools commonly used by 'hackers' could be easily used to demonstrate certain networking principles as well as security at the same time. Consider, for example, an assignment in which students run a sniffer to observe data traffic. This assignment could be used to demonstrate the concept of packets and to illustrate the many different types of network traffic that exist on a line. At the same time the students would be able to see how vulnerable network traffic is and how easy it is to obtain valuable information such as passwords if an individual has the ability to run such a program. Obviously the lab environment must exist to allow students to run such a package without impacting other courses and individuals. If such an environment exists, another lab that could be used to illustrate similar principles would involve the use of software designed to perform IP spoofing or session hijacking. Either of these would illustrate both the function of network protocols as well as their vulnerability to misuse. Other topics for research/discussion would include firewalls, cryptography (public key and key escrow/recovery in particular), intrusion detection, and traffic analysis.

2.8 Software Engineering

The ultimate goal should be to educate students so that when they are involved in the design of software they always consider the security implications of what it is they are doing. A natural place to stress this is in a software engineering course. Students could be given assignments, for example, to develop security specifications and test plans for software systems. They could also be tasked with the analysis of the security criteria for a specific application or software system or they could be responsible for determining the security policy for a system. The most important point here is that no matter what the assignment given to them in a software engineering course, the students must be held accountable for insuring that the security of the system is considered.

3. SEPARATE SECURITY COURSES

Even if security is successfully integrated throughout a department's computer science curriculum, there will still be a need for separate or specialized courses in security. For example, while cryptography is mentioned at numerous places in the curriculum, the topic is rigorous enough that a separate course on this topic alone could easily be supported. This would allow individuals who want to specialize in computer security to obtain extra training in the subject. There are several other courses that could be offered at both the graduate and undergraduate levels to instruct those wishing to know more about security. A general course on computer security itself would allow students the opportunity to delve deeper into the many issues found scattered throughout the curriculum. More specialized courses in areas such as intrusion detection/prevention might be more applicable at the graduate level but would provide instruction in this extremely important area. So too would courses on firewalls or system administration. Specialized courses such as these would provide an opportunity for examination of tools such as COPS, Crack, SATAN, Back Orifice and NetLog and the methodology used in them. They would also allow exploration of the 'hacker' community and mindset and examination of software packages that have been created to exploit vulnerabilities found in systems (such as *winnuke*, *smurf*, and *ping*).

One particularly useful semester-long project used at the United States Air Force Academy was to break students into teams of two and assign each team a machine in the lab. These machines were dedicated to the class for the semester and students were required to load the operating system and maintain a certain minimal set of services. Each team was also tasked with doing whatever was required to protect their system while at the same time attempting to gain access to the other teams systems. At the end of the semester each team briefed the class on what they learned as well as the methods they used to gain access to the systems controlled by the other teams. Many teams often had no idea that their system had been compromised. This project not only provided many useful lessons in security, it also provided the students with valuable system administration experience.

4. CONCLUSION

Few, if any, would argue against the necessity to provide instruction on computer security in both computer science and related majors. The question, however, is how best to approach this instruction. Adding additional required courses is not possible in most programs so is not considered a viable option. The solution then is to incorporate security throughout the curriculum. This can be done without detracting from current course content by carefully selecting labs, papers, and other assignments to include security related concerns. If chosen correctly, the assignments can illustrate both core computer science topics as well as security issues. By including security in all applicable courses, students will come to realize that security should not be an afterthought but is a cornerstone in good software design.

5. REFERENCES

Denning and Denning, Internet Besieged: Countering Cyberspace Scofflaws, Addison Wesley, Reading, 1998.

Garfinkel and Spafford, Web Security and Commerce, O'Reilly & Associates, Inc., Cambridge, 1997.

Gasser, Morrie, Building a Secure Computer System, Van Nostrand Reinhold, New York, 1988.

White, Fisch and Pooch, Computer System and Network Security, CRC Press, Boca Raton, 1996.