Information Security and Computer Systems: An Integrated Approach

Mark A. Holliday

Dept. of Mathematics and Computer Science
Western Carolina University
Cullowhee, NC 28723
01-828-227-3951

holliday@wcu.edu

William C. Kreahling

Dept. of Mathematics and Computer Science
Western Carolina University
Cullowhee, NC 28723
01-828-227-3944

wck@cs.wcu.edu

ABSTRACT

As part of a major redesign of our computer science curriculum we are developing an Information Security option. Our approach highlights the many topics in information security that build upon concepts the students will already have seen in their computer systems courses, especially courses on internet protocols and operating systems. In this paper, we describe this integrated approach to information security and computer systems.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection; C.2.4 [**Computer-Communications Networks**]: Distributed Systems; E.3 [**Data Encryption**]

General Terms

Algorithms, Measurement, Design, Security, Legal Aspects.

Keywords

Information security, operating systems, internet protocols, cryptography, malicious software

1.INTRODUCTION

We recently decided to redesign the curriculum for the B.S. in Computer Science at Western Carolina University. A major goal of the redesign is to provide more choices for the student, while at the same time having the choices be organized around a coherent theme. The resulting design has a small (25 credit hours) core of computer science courses that all computer science majors take and a set of options. Each option is a substantial set of computer science courses (15 credit hours) that are related. One of the options will be Information Security. This paper addresses the design choice made for that option given the framework of the rest of the curriculum. A key theme of our design is our view that the best approach for incorporating Information Security into a computer science major is to have a technical focus that extends a solid understanding of related concepts in computer systems.

Information Security Curriculum Development (InfoSecCD) Conference'06, September 23-24, 2006, Kennesaw, GA, USA.

The next section provides an overview of the new design of our computer science major. This overview makes clear the constraints we face on the number and types of information security courses we can add. The overview also identifies the background we can assume most students have had before they start the Information Security option. Section Three describes our first attempt and final solution for how to structure the dependencies between the computer systems courses and the information security courses within the Information Security option. Section Four addresses the content of our first Information Security course and the connections to our Internet Protocols course. Section Five similarly addresses the issues involving the content of our second Information Security course and its relationship with our Operating Systems course. We conclude in Section Six.

2.THE CURRICULUM FRAMEWORK

To understand our design choices for our Information Security option it is necessary to understand the framework in which it exists, the overall curriculum for the B.S. in Computer Science degree at Western Carolina University. The starting points for the curriculum are two sets of guidelines:

- ACM and the IEEE Computer Society have jointly developed a set of curriculum guidelines for a Computer Science major [1].
- The accreditation body for Computer Science, the ABET-Computing Accreditation Commission (CAC), has developed a set of criteria for accrediting a computer science program [2] that covers all the aspects of a program including the curriculum.

The 2001 ACM/IEEE curriculum guidelines for the computer science major have a key theme: computer science has broadened to such an extent that the best approach is to have a small core of required computer science courses combined with a sizable number of computer science electives. That report proposed a core consisting of 280 lecture hours with the assumption that a normal three credit hour semester long (15 week) course has 40 lecture hours. Thus, 280 lecture hours is equivalent to seven three-credit-hour courses; in other words, 21 credit hours.

The 2001 report also identifies a Computer Science Body of Knowledge including the subset of that Body of Knowledge that should be in the core for any computer science curriculum. The subset that is in the core as defined by the 2001 report influenced what courses we chose to include in our core.

The ABET-CAC accreditation criteria while briefer than the 2001 ACM/IEEE report, also cover many topics in addition to the curriculum. However, those criteria have a number of similarities with the 2001 report. In particular, the accreditation criteria divide the computer science courses into a core and a set of computer science electives. Also, the topics in the core are similar to those in the 2001 report. The topics that should be in the core according to the accreditation guidelines are:

"IV-6. The core materials must provide basic coverage of algorithms, data structures, software design, concepts of programming languages, and computer organization and architecture." [2, page 3]

Based on these two sets of guidelines we developed a core for our major that consists of 25 credit hours. Figure 1 shows the dependency graph for our core.

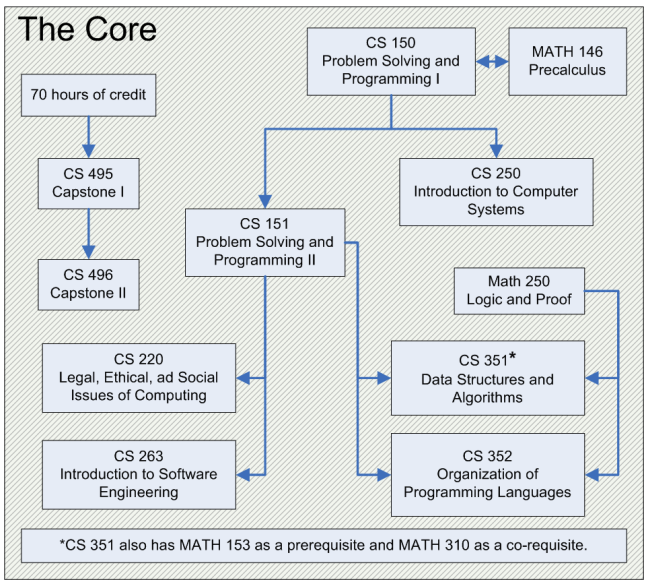

Figure 1: The core for the new design of the B.S. in Computer Science at Western Carolina University.

The core includes a two course programming sequence in Java that includes many data structures such as linked lists, stacks, queues, and search trees. There is a course introducing software engineering and a course introducing computer systems. The computer systems course includes a substantial section on computer organization and architecture, but also provides an initial coverage of language translation and operating systems. At a somewhat more advanced level we offer a course that completes the coverage of data structures and provides a rigorous introduction to algorithm design and analysis. At that same level we have an organization of programming languages course. A one-credit hour course on legal, social, and ethical issues and a two-semester capstone sequence complete the core.

The ABET-CAC accreditation criteria identify the total number of semester hours of computer science and of mathematics and sciences.

"IV-1. The curriculum must contain at least 40 semester hours of up-to-date study of computer science topics.

IV-2. The curriculum must contain at least 30 semester hours in mathematics and science." [2, page 3]

We designed a major that also satisfies these requirements. The mathematics and sciences required total 31 semester hours in the form of 14 semester hours of science courses and 17 semester hours of mathematics (two semesters of calculus, a statistics, course, a discrete mathematics course, and a logic and proof course). For the computer science courses we created several options with each option consisting of 15 semester hours (five courses) of computer science electives. Because the core has 25 semester hours, the computer science total is 40 semester hours.

The courses that are required for the Information Security option have our two course programming sequence (CS 150 and CS 151), our introduction to computer systems courses (CS 250), and our software engineering course (CS 263) as prerequisites that are in the core. Thus, a student starting the Information Security option should have a solid background in object-oriented programming and software engineering, and be comfortable with assembly language, the key concepts in the hardware implementation of the assembly language interface, language translation, and operating systems. Due to advising and scheduling, it is very likely that before starting the courses in the option the student will have also completed the legal, ethical, and social issues course that is in the core as well as at least one of the two semesters of calculus, and the logic and proof course.

All computer science courses must be completed with a grade of at least a C in order for the student to graduate. It is true, however, that a student may retake a passed prerequisite course in order to earn the minimum grade of C after starting the Information Security option. An alternative approach would be to require the student to complete every course in the core with at least a grade of C before starting the courses in any option. We rejected this approach due to the likelihood that many students might not be able to graduate within four years.

The above framework also presents some constraints on the design of the Information Security option. One constraint is that due to the number of hours required at the university-level for Liberal Studies and general electives, we could not increase the number of computer science semester hours without exceeding the maximum total of 120 semester hours. Thus, 15 semester hours (5 courses) is the maximum number of hours in an option.

A second constraint is a result of our goal of providing more course choices for our students. To offer more course choices for our students with the same number of faculty, individual courses will have to be offered less often. In particular, since we are a relatively small computer science program, all the computer science electives (that is, the courses in the options) will be offered once every two years. This restricts what prerequisites we place on the information security courses given that we are strongly encouraged to ensure that the student is able to graduate within four years.

3. THE OPTION'S STRUCTURE

We contend that a student completing a Computer Science major should have developed a strong technical understanding of how computer systems work from the hardware level through language translation, operating systems to database systems and computer

networking. Such an understanding makes concrete the use of data structures and algorithms and enhances software development skills through the projects completed to demonstrate understanding.

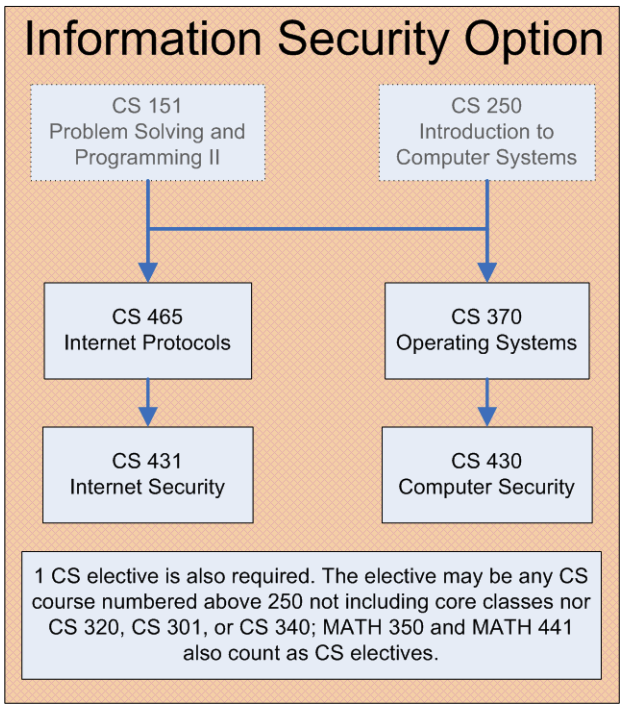

Figure 2: Our first attempt at the structure of the Information Security option.

Given this perspective, Information Security is a natural option for a Computer Science major since the concepts covered are so closely connected to the concepts the students have seen in their computer systems courses. In fact, it is not clear to us how a student could develop proficiency in Information Security without a strong background in computer systems. The problem, of course, is the option can only require five courses beyond the core. To be consistent with our goal of giving the students more choices at least one of the five courses should be an arbitrary computer science elective so that the student has some flexibility. The computer science electives we often offer besides those in the Information Security option include Database Management Systems, Computer Architecture, Advanced Software Development, Computer Graphics, Theory of Computation, and Numerical Analysis, Thus, in the Information Security option, only four of the five courses specified have to be information security and computer systems courses.

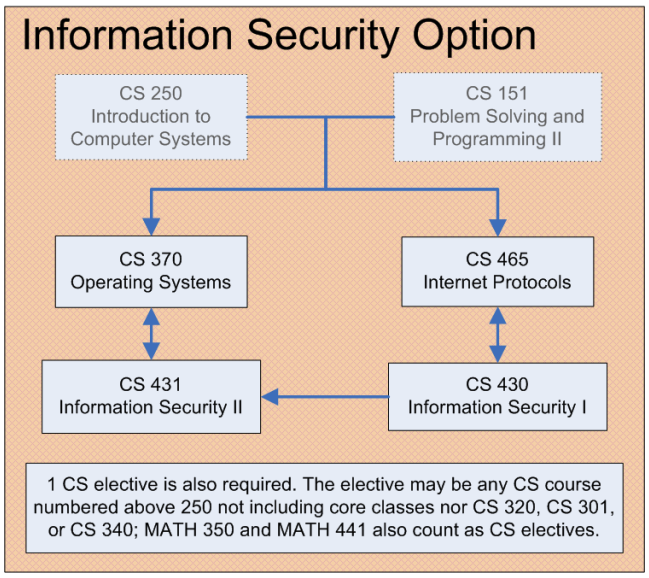

Figure 3: Our final structure for the Information Security option.

The solution to this challenge given our curriculum framework is now described. We offer four computer systems courses: Computer Architecture, Operating Systems, Database Management Systems, and Internet Protocols. Information security is an issue in all four of these areas. However, a strong case can be made that the two most relevant systems areas are covered by the Operating Systems course and the Internet Protocols course. Many security issues involve only a single computer and do involve topics covered in an operating systems course. Even more importantly, security protocols are a fundamental part of computer networking

Our first attempt at designing an Information Security option is shown in Figure 2. We planned a Computer Security course, CS 430, that has the Operating Systems course, CS 370, as its prerequisite. The title Computer Security was planned for the CS 430 course since it would focus on single computer issues and thus is closely related to the Operating Systems course. We planned an Internet Security course, CS 431, that has that Internet Protocols course, CS 465, as its prerequisite. The title Internet Security was planned for the CS 431 course since it would focus on network security and thus is closely related to the Internet Protocols course.

Even though each course is only offered once every two years, the student would be able to complete the option in two years regardless of which year the student starts the option. In other words, the student could take the two course sequence Operating Systems/Computer Security before or after the two course sequence Internet Protocols/Internet Security.

It became clear that this approach would not succeed as we worked further on the content of the two security courses. The problem is that there is a basic set of concepts on cryptography (symmetric key and public key) and its uses for confidentiality, authentication, message integrity and non-repudiation that needs to be covered when studying the single computer case as well as in the networked case. Thus, there has to be a dependency between the two information security courses, in addition to the dependency with the Operating Systems course and the Internet

Protocols course. As a result of the dependency between the two information security courses, in the final version of the Information Security option we changed the titles of the two security courses to Information Security I (for CS 430) and Information Security II (for CS 431). The new titles made clear the dependency between those two courses.

The approach we then developed is shown in Figure 3. There are several key points to this new approach. The first key point involves organization of course material. We think of the Internet Protocols course as being a more *basic* systems course than the Operating Systems course. This idea might seem counter-intuitive. In teaching computer systems, it is natural to adopt a "bottoms-up" approach that starts with the lower levels of a single computer (computer organization and then operating systems) and finish with the issues involving networking computers together. However, with respect to security, we argue that the more natural evolution starts with the network and then examines issues involving a single computer. The reason is the pervasiveness of cryptographic techniques in information security and the more natural introduction of those concepts in a computer networking course.

The second key point involves the problem that the additional dependency between the two security courses makes offering of CS elective courses only once every two years problematic. The solution is to use co-requisites so that all four courses in the option are offered in a single year. In the Fall semester we offer the Internet Protocols course and the Information Security I course. In the following Spring semester we offer the Operating Systems course and the Information Security II course. During the second year of the cycle we offer all of our other CS electives.

Thus, a student needs only two years to complete the option regardless of which year the student finishes the courses needed before the option. Furthermore, using co-requisites instead of pre-requisites is preferable since the material in the companion computer systems course is fresh in the students' minds as they are taking the security course. However, the scheduling of material within each course has to be carefully planned so that preliminary concepts needed in the computer systems course are covered before they are used in the companion information security course.

4.INFORMATION SECURITY I

As discussed above, the design choice for the security courses is to introduce cryptography and its application to the properties of secure communication (that is, confidentiality, authentication, message integrity, non-repudiation, availability, access control) as early as possible. This implies that the first security course will be taken during the same semester as the Internet Protocols course, since network examples are the most natural examples of these uses of cryptography.

The textbook for our internet protocols course is Kurose and Ross [4] which fortunately includes an excellent chapter on security in computer networks. Thus, although the security course has its own textbook, the chapter in Kurose and Ross is a useful supplement for the students. Tentatively, we are planning on using Kaufman, Perlman, and Speciner [3] as the textbook for the Information Security I course.

Table 1 shows the design of the first security course and the Internet Protocols course that students will take concurrently. After identifying the properties of secure communication, cryptography is introduced, both symmetric key and public key. It is then shown how cryptography can be used for authentication which involves introducing example attacks including the playback attack (and thus, the use of a nonce) and the man-in-the-middle attack. Cryptography is then used for message integrity and non-repudiation. The latter uses necessitate the introduction of digital signatures, message digests, and hash function algorithms.

The use of cryptography in all these forms, directly relates to the Internet Protocols course since these are all network protocols. A further connection is through hash functions and message digests. Around this point the concept of a hash function is being introduced in the Internet Protocols course for error-detection via checksums in UDP, TCP, and IP. Cyclic redundancy checks in Ethernet are another example of using a hash function to detect errors.

Table 1. Content and Dependencies in the Fall courses

Internet Protocols	Information Security I
Key cross-layer concepts (e.g. encapsulation, multiplexing, fragmentation)	Properties of secure communication; Cryptography (symmetric key, public key);
Application Layer (e.g. HTTP, DNS)	use in authentication; use in integrity and non-repudiation
Transport Layer (TCP, UDP, bandwidth versus propagation delay, reliable data transfer)	Trusted intermediaries, key distribution, and certification; Access control: firewalls
Network Layer (IP, ICMP, routing algorithms (OSPF, RIP, BGP))	Attacks and counter-measures
Link Layer (Ethernet, 802.11, error detection)	Security examples at each layer

The next section of Information Security I addresses how a trusted intermediary can be used to establish a shared key for symmetric key cryptography and to securely obtain a public key in public key cryptography. Kerberos and the certificate authority, SimpleCA, that comes with the Globus toolkit for grid computing are used as examples. We have Globus installed on our local cluster. That SimpleCA is implemented as a web service relates to the Internet Protocols course. The Internet Protocols course first introduced inter-process communication (IPC) using socket programming. Later in the semester it introduces Java remote method invocation (RMI) and web services as higher-level ways of providing IPC.

Firewalls are introduced as a way of providing the access control property of secure communications. A table of packet-filtering rules looks quite similar to an IP routing table which the students are seeing in the Internet Protocols course about this time. The use of IP addresses, UDP source and destination ports, ICMP message type, and other TCP header bits (such as SYN and ACK bits) in packet-filtering reinforces those concepts first seen in the networking course.

Having covered the general approaches to ensuring the properties of secure communications, the security course looks at security from a different angle: example attacks and how to stop them. Attacks discussed include mapping, port scanning, packet sniffing, spoofing, hijacking an ongoing connection, and denial-of-service (DoS) attacks (such as SYN flooding). The networking

course should help to make these clear. For example, the use of the SYN bit in the TCP header as part of connection-establishment has already been covered in the Internet Protocols course.

The security course then takes a third approach by looking at example ways of providing security at each of the layers of the protocol stack. The application layer example is secure email and PGP. For the transport layer the example is Secure Socket Layer (SSL) and Transport Layer Security (TLS). IPsec is the example at the network layer. The IEEE 802.11's security mechanism, Wired Equivalent Privacy (WEP), is the example at the link layer. All of these layers have been covered in the Internet Protocols course at this point. For example, aspects of the IEEE 802.11 family of protocols are covered as examples of link layer protocols.

The Internet Protocols course has a number of network programming projects involving sockets, remote method invocation (RMI), and web services. The Information Security I course similarly will have a number of security programming projects using the Java security API.

5.INFORMATION SECURITY II

Given that the Information Security I course is the student's first security course and since it is taking place during the same semester as the Internet Protocols course, it is quite clear what the content of that course and sequencing of that course should be. On the other hand, there is significantly more flexibility in the Information Security II course.

There are a number of security topics that are based on the concepts introduced in an Operating Systems course, so those topics belong in the Information Security II. However, there are also a number of topics in network security that warrant further coverage. Our result is a mixture of those two strands. Table 2 shows the design of the second security course and the Operating Systems course that students will take concurrently.

Our Operating Systems course begins with a set of basic concepts: the system call interface being the abstract machine defined and implemented by the operating system kernel, the abstractions provided by the system call interface (processes, files, IPC (signals, pipes, sockets), synchronization, threads and so on), the organization of the process address space, exception handling (for both system call execution and other exceptions), the need for multiprogramming because of how long input/output can take, context switching, key parts of the kernel (such as, process control blocks), the interrupt vector table, protection modes (user and kernel modes), the trap instruction, the PC and PSW registers, file permission modes, different user accounts having different permissions, and the permissions of user root, the setuid bit and a normal user being able to execute a particular file as root.

While these concepts are being covered in the Operating Systems course, a natural companion topic in the security course would be techniques for studying the binary file of a process to detect if it is malicious software. This involves software reverse engineering and requires an understanding of assembly language and the organization of the address space of a process. The students have already completed CS 250, our computer systems course in the core, which among other topics has covered assembly language programming. So the students are ready for this topic. We discuss how software reverse engineering can be used to identify malicious software such as viruses, worms, trojan horses, spyware, and programs with trapdoors.

Being able to reverse engineer a binary leads into a discussion of legal issues such as the U.S. Digital Millennium Copyright Act and copyright law.

The second extension of reverse engineering examines exploits that compromise program security. These exploits closely relate to the process address space since they involve issues such as the protection mode code is executing in (user mode or kernel mode), whether a memory segment is read-only (for example, the code segment) or read-write, and the layout of the stack and the heap segments. Buffer overflow exploits often target the stack and the heap. Those example buffer overflow exploits lead into other buffer overflow exploits involving space for arrays and strings and that C and C++ do not provide automatic bounds checking. We introduce defenses by code analysis (static analysis or at runtime) and by the compiler.

The above discussion of program security covers exploits that exist for arbitrary programs. Another extension is to look at program security for specific classes of programs. Such program classes include web browsers and cookies, email viruses, Microsoft Word viruses, and boot sector infections. The different types of anti-virus software are covered.

Table 2. Content and Dependencies in the Spring courses

Operating Systems	Information Security II
Basic Concepts; system call interface; exception handling, the process address space	Software reverse engineering and legal issues
Process Management (cpu scheduling, synchronization, deadlock)	Program security (general programs and specific program classes)
Memory Management (demand paging, virtual memory)	Security administration: prevention, detection, and response
Input/Output and File Systems	International Standards for Information Assurance

The remainder of the Information Security II course addresses security administration: prevention, detection, and response. Prevention starts with risk assessment combining both penetration testing and techniques to harden the operating system (e.g. changing default ports, changing the encryption level, anti-virus software, enabling firewalls, turning on logging, user account management, and physical security). For detection there are network monitoring (e.g. to detect intrusions), computer forensics, and log monitoring. For response we consider ways to recover from a security breach including system backups.

The last topic introduces the international standards that have been developed to address information assurance including the names of the documents and the terminology that the documents use (such as the security classes in the Orange Book and the functional requirement classes in the Common Criteria (CC) Project).

6.CONCLUSIONS

Information security is an increasingly important topic for any academic discipline within information technology. Its increasing importance justifies and requires significant curriculum changes across all these academic disciplines. This paper focuses on one particular case and how one institution is attempting to accommodate these changes. The particular case is the curriculum for the B.S. in Computer Science degree in a small computer science program.

ACM/IEEE curriculum guidelines and ABET accreditation criteria motivate a framework of a small core complemented by several options with each option consisting of five courses. The importance we place on technical understanding of computer system concepts along with the close connections of computer system concepts and information security concepts imply that the option must have a significant number of both types of courses. Being a small computer science program requires that the courses in the option be offered only once every two years. We have explained the choices we have made that accommodate these constraints.

The structural choices involve selecting four courses for the option: Internet Protocols, Operating Systems, Information Security I, and Information Security II. A combination of prerequisite and co-requisite constraints between those courses ensures that topics are covered in a natural, intuitive order and that students will have no difficulty completing the option within a two-year cycle.

The content choices for the material within the four courses allow us to cover a significant amount of material on information security. Those choices also allow the computer systems courses and the information security courses to reinforce each other.

7.ACKNOWLEDGMENTS

Our thanks to Sam Daoud for drawing the dependency graph figures.

8.REFERENCES

[1] *ACM/IEEE 2001 Task Force. Computing Curricula 2001. Computer Science*. IEEE Computer Society Press and ACM Press. December 2001.

[2] *Criteria for Accrediting Computing Programs: Effective for Evaluations during the 2006-2007 Accreditation Cycle.* ABET-Computing Accreditation Commission, Baltimore, MD.

[3] Kaufman, C., Perlman, R., and Speciner, M., *Network Security: Private Communication in a Public World, Second Edition*, Prentice-Hall, 2003.

[4] Kurose, J.F., and Ross, K.W.*, Computer Networking, Third Edition*, Addison-Wesley, 2005.

[5] Lindner, F., Software Security is Software Reliability, *Communications of the ACM*, June 2006, vol. 49, no. 6, pp. 57-61.