Project Summary Page

Project Title: Towards a Security-Aware Undergraduate Computer Science Curriculum at UMBC

Date: March 19, 2003

Investigator Names:

- Brian Roberts, Graduate Student @ UMBC, roberts2@umbc.edu
- Doug Cress, Graduate Student @ UMBC, cress1@umbc.edu
- John Simmons, Graduate Student @ UMBC, js5@umbc.edu

Information Assurance Area: secure software development education

Keywords: undergraduate education, security modules, secure programming, computer science curriculum

Research Question: How would one incorporate basic, class-appropriate, security material into core, undergraduate, programming classes at UMBC?

Answer/Project goal: We will develop a class-specific, security module that can be easily integrated into the existing syllabus of Computer Science I for majors (CMSC-201), Computer Science II for majors (CMSC-202), & Data Structures (CMSC-341).

Assigned Responsibilities: Brian will be primarily responsible for developing the module for CMSC-201, John will be primarily responsible for developing the module for CMSC-202, and Doug will be primarily responsible for developing the module for CMSC-341.

Total Budget: $10,854.00

Deliverables:

- 3 curriculum modules, one for each class, comprised of a set of PowerPoint slides providing basic background and an example germane to the subject matter of the class.
- A paper discussing the development of the modules, trade-offs weighed and reactions to the modules from the instructors of the classes.
- A PowerPoint presentation describing the results of our research.

Executive Summary

Project Title: Towards a Security-Aware Undergraduate Computer Science Curriculum at UMBC
Date: March 19, 2003
Investigator Names:

- Brian Roberts, Graduate Student @ UMBC, roberts2@umbc.edu
- Doug Cress, Graduate Student @ UMBC, cress1@umbc.edu
- John Simmons, Graduate Student @ UMBC, js5@umbc.edu

Keywords: undergraduate education, security modules, secure programming, computer science curriculum

As the usefulness and ubiquity of information technology (IT) has increased, the deployment of poorly-written insecure code has exploded. Hundreds of vulnerabilities are revealed by both security defenders and malicious attackers each year. This ever widening gap between good, secure, software development supposedly taught in schools, and the actual bug-ridden code that is deployed will continue to plague our society and endanger our future.

In order to address this critical problem, our team has taken on the task of analyzing the current curriculum of the three foundational computer science classes required of all students seeking to graduate with a Bachelor's of Computer Science from UMBC. These classes are CMSC-201 Computer Science I for majors, CMSC-202 Computer Science II for majors, and CMSC-341 Data Structures. After we've examined the topics covered by the classes, we will develop specific germane examples and teaching tools that will assist the students to understand where they can start to write more secure code. Additionally we'll describe the dangers that poor, insecure software development can lead to and point out how they personally, could contribute to the destruction or un-authorized access of important data on the Internet.

After we've developed our course modules, designed to encompass a single lecture or additionally condensable to a half lecture, we will meet with current instructors of each of the classes to see how they would use our material to instruct their students and to seek any additional suggestions they might have for improvement. Once we've combined all their suggestions into our work we will present the modules to the department for possible inclusion into the base CS curriculum.

Our work is both novel and significant. Novel in that such Information Assurance based education is currently completely lacking from these classes. Any addition of such principals will be new and practical. Significant because of the critical nature that security does play in Information Technology. Hopefully our work will begin to penetrate the general malaise currently directed towards secure programming at the undergraduate level.

Motivation

Security is an increasingly important aspect of Computer Science, as more and more reliance is placed on information systems and the Internet. In UMBC's undergraduate curriculum, security is a topic only covered in elective classes; thus it is possible and common for students to graduate with degrees in Computer Science without ever receiving instruction on security issues. Through the personal experience of the researchers, it is evident that this problem is not at all unique to UMBC. If information systems are to be resistant to errors and malicious code, those who design the systems need to be aware of security issues.

We feel that an important step towards this goal is to expose Computer Science students to security issues in their coursework. It is unnecessary to reconstruct an entire curriculum to this end; a set of learning modules could be included in existing courses to expose students to this material. Ideally, one module could be tailor-made for almost every course in the curriculum. The module would emphasize the security aspects most closely related to the course, perhaps exploring the underlying concepts of security principles learned in earlier courses. To limit the scope of this project, we will only be developing modules for three courses at UMBC: CMSC 201 (Computer Science I for majors), CMSC 202 (Computer Science II for majors), and CMSC 341 (Data Structures). We feel that since these three courses comprise the bulk of each student's programming instruction, they are an excellent place to begin teaching secure programming principles and practices.

Critical Review of Previous Work

We begin our review of previous work by examining the current core programming curriculum at UMBC. Upon entering the computer science degree program at UMBC each student is required to complete a series of courses introducing them to computer programming. These three courses are CMSC 201 (Computer Science I for majors), CMSC202 (Computer Science II for majors), and CMSC 341 (Data Structures). These courses work together to teach the novice computer science student the fundamental concepts of programming in the C language, advanced and object oriented programming in C++ and programming data structures. Each course is designed specifically for a certain level of programming expertise and each course builds upon the previous course material (requiring that course as a prerequisite) to give students a thorough understanding of structured programming concepts.

While these courses strive to teach proper coding standards, language functionality, and basic problem solving, they mention very little about secure programming practices. The general approach is to instruct students on what behaviors or functionality of the language to avoid, without giving a thorough explanation of why these behaviors are

dangerous. We feel that in order for students to fully understand the implications of security (and the lack thereof) it is important to focus on these topics more thoroughly and present a more technical and in depth examination of the concepts (within the scope of expectations for the course the module is being used in).

Since 1999 the National Security Agency has designated thirty-six universities as Centers of Academic Excellence in Information Assurance. In accordance with this designation these universities must adhere to a list of 10 qualifications including an overall curriculum mapped to various National Security Telecommunications and Information Systems Security Standards (NSTISSI). The foremost of which, number 4011, is the "National Training Standard for Information Systems Security (INFOSEC) Professionals," dated June 20, 1994. This standard is meant to be viewed as a curriculum to teach students how to be INFOSEC professionals. Sadly this standard does not quite cut the mustard. It primarily focuses on policy and general Automated Information Systems (AIS) management and planning. Though these are important components for a graduate level course, this level of abstraction does not contain sufficient detail for teaching undergraduate students how to code in a secure manner. Since all AIS would be worthless hunks of plastic and metal without software to run on them, we believe that secure software development should be addressed at every step down the path towards a bachelor's in computer science.

In June of 2002 the second annual Information Assurance Curriculum development workshop was held, sponsored by Purdue University. Melissa Dark and Jim Davis chronicled the experience in a paper called "Report on Information Assurance Curriculum Development." The workshop split up the participants into two groups one focused on graduate level studies and one on undergraduate studies. The groups sought to classify the knowledge that students needed to attain into three domains: Declarative (memorization of facts & figures), Application (ability to use declarative knowledge previously acquired) and Synthesis (to create new solutions from existing knowledge). Out of the fourteen broad subject headings only, one was dedicated to secure software engineering practices. The four subjects listed (security of large software systems, programming language issues, software engineering techniques and security issues) were good broad observations of the types of skills that the students should learn. However detail was again lacking. A description of exactly what portions of an IA curriculum a student should be responsible for at a particular stage in his/her education was missing. Inclusion of specific pitfalls and how to avoid them in software engineering were left out. We hope that our work can start with these areas and provide specific solutions relative to the UMBC student, but generic enough in scope to allow other universities to gain from our research.

The paper [YA01] examines the need for security to be taught in undergraduate curriculums. It gives many reasons why this is needed, and provides several outside sources of material which may be helpful to those designing security lessons. Integration of security topics into an existing curriculum is examined in detail, with suggestions for each course, though no details are provided. This paper also spends some time discussing

the obstacles and issues facing anyone trying to enact real changes to a university's curriculum.

In [VA96], similar topics are covered in a much more abbreviated fashion. Specific recommendations are made for course content, but not as many courses are considered as in [YA01]. The courses treated are operating systems, database, software engineering, computer networking, and artificial intelligence. It is interesting that for all its concern with security topics in advanced courses, this paper does not touch on including security instruction in introductory programming courses.

The paper [MU02] is a brief introduction to a panel discussion that was held at the SIGCE '02 conference concerning the integration of security material into existing Computer Science courses. This matches almost perfectly with our project. Unfortunately, this document only contains position statements from each of the panel members, and a transcript of the discussion does not seem to be available.

Specific Aims

- Develop teaching modules for the three core programming courses that comprise the UMBC undergraduate programming curriculum.
- Introduce secure programming practices via these teaching modules.
- Distribute material accordingly between teaching modules to reflect the experience level of students in each of the core classes.
- Build on each module to provide greater depth and technical detail in each subsequent module.
- Provide opportunities for adaptation of materials as core curriculums change.

Plan

In order to meet our specified aims, we intend to study the current curriculum for each of these classes and determine the extent of security concepts covered in the material. We will then isolate areas where secure programming practices can be best added. Once these areas are identified, we will develop a series of exercises for the purpose of familiarizing students with secure programming practices.

Initially we will begin each of the three modules by reviewing any prerequisite concepts that are required to understand the topics in the module, while most of these required topics are already covered in the existing curriculum, they will be refreshed and put into the context of secure programming. Following the initial review process, students will begin learning common programming errors that they might routinely make during the course and the security implications of these errors. Case studies will be used to illustrate the dangers of insecure programming practices. This process will lead to the

identification of basic buffer overflow situations in code and a thorough examination of examples of insecure code. Students will also learn the importance of topics such as input verification and the inherent weaknesses of the language they are programming in. Eventually the module will culminate in the incorporation of these secure programming practices into a hands-on programming exercise.

Deliverables

- 3 curriculum modules, one for each class, comprised of a set of PowerPoint slides providing basic background and an example germane to the subject matter of the class.
- A paper discussing the development of the modules, trade-offs weighed and reactions to the modules from the instructors of the classes.
- A PowerPoint presentation describing the results of our research.

Issues

The most challenging aspect of designing these learning modules will be deciding what content needs to be included, and what depth of coverage is appropriate for each module. If our learning modules are to be practical for integration into existing courses, their scope needs to be limited to one or two lectures per course. This puts quite a limitation on the amount of material we can include in any one module, and we will need to decide which topics are most important. Also, the material in each module must be presented at a level which the students of that course can understand. This concern is especially important for lower level courses.

Tips and guides for secure programming practices can already be found in numerous books and web sites. Our project is different from these in that it will be designed specifically for use in an academic setting. Thus it is very important that the modules produced be practical for use in their respective courses. To evaluate the practicality of our modules, we feel that the best approach is to seek the opinions of faculty who teach the target courses. The two questions we feel best fit our feedback needs are the following: is this module appropriate for the students in your class, and could you reasonably integrate this module into your teaching? Receiving good feedback of this sort is critical for this project to be a success.

Bibliography

[AN96] Anonymous, “A Lab Engineer’s Check List for Writing Secure Unix Code”, May 1996. ftp://ftp.auscert.org.au/pub/auscert/papers/secure_programming_checklist

[DD] M. Dark, J. Davis, "Report on Information Assurance Curriculum Development", June 2002.
http://www.cerias.purdue.edu/education/post_secondary_education/undergrad_and_grad/curriculum_development/information_assurance/report_info_assurance_cur_dev.pdf

[JMC] J. McConnell, "National Training Standard for Information Systems Security (INFOSEC) Professionals—NSTISSI No. 4011", June 1994.
http://www.nstissc.gov/Assets/pdf/4011.pdf

[MU02] Mullins, "Panel on Integrating Security Concepts into Existing Computer Courses", March 2002, ACM SIGCE Bulletin, Vol. 34, p. 365

[NCSA] NCSA, "NCSA Secure Programming Guidelines", no date.
http://archive.ncsa.uiuc.edu/General/Grid/ACES/security/programming/

[VA00] Richard Vaughn, "Application of Security to the Computer Science Classroom", March 2000, AMD SIGCE Bulletin, v. 32, p. 90

[YA01] T. Andrew Yang, "Computer Security and Impact on Computer Education", May 2001, ACM Journal of Computing in Small Colleges vol. 16, p. 233

Biographical Sketches

Brian Roberts: Brian is a graduate student at UMBC interested in Information Assurance. He graduated from Towson University with a B.S. in Computer and Information Systems. Following graduation he worked as a systems administrator and consultant.

Douglas Cress: Doug is currently a graduate student at UMBC studying the implications of network scanning on computer network attack at high bandwidth. He graduated with honors from James Madison University, with a Bachelor's in Computer Science. Since then he has worked for the Department of Defense.

John Simons: John is a graduate student in the Master's degree program at UMBC. He received a Bachelor's degree majoring in Computer Science and Mathematics from Frostburg State University.

Schedule

March 19	Proposal
March 21	Evaluation of a peer proposal by email
March 28	Develop list of topics
April 2	Progress report
April 4	Develop coverage goals
April 7	Finalize midterm
April 9	Midterm delivery
April 16	Interview Faculty
April 21	Pre-draft of paper and presentation
April 23	Complete draft report and draft presentation for review
April 30	Referee report of a peer project
April 30	Oral presentations begin
May 7	Final report

Budget

Budget

Direct Costs	
Researcher Wages	$8,000.00
Printing and Media Fees	$100.00
Subtotal:	$8,100.00
Indirect Costs	
Overhead	$2,754.00
Totals	$10,854.00

Research Conference

The ACM's Special Interest Group in Computer Science Education (SIGCSE) holds an annual conference which fits very well with the topic of our project. The next one is SIGCSE '03 in Reno, NV, and can be found at: http://www.csis.gvsu.edu/sigcse2003/