Secure Software Engineering

Samuel T. Redwine, Jr.
James Madison University
701 Carrier Drive
Harrisonburg, VA 22807 USA

Abstract – **In the fall of 2004, James Madison University began offering a two years Masters degree in Secure Software Engineering. Among its required courses are four secure software engineering courses, an introduction to security, two network courses one emphasizing security, and several traditional computer science courses. The four software engineering courses include an initial project course that covers the entire lifecycle and a three semester sequence that more expansively repeats the lifecycle. This article describes the experience and some of what has been learned from the offerings emphasizing the initial course.**

I. INTRODUCTION

In 2003 motivated by the desire to increase enrollments and use faculty expertise, the Computer Science Department at James Madison University decided to offer a Masters degree in Secure Software Engineering.[1] While the goal of increasing enrollments has not been fully met, interesting and valuable experience and lessons have been gained in teaching (and learning) secure software engineering that are worth sharing.

Required courses include four secure software engineering courses, an introduction to security, two network courses with one emphasizing security, and several traditional computer science courses. The four software engineering courses begin with an initial project course that covers the entire lifecycle, and this is followed by a three semester sequence that more slowly and with more detail repeats the lifecycle.

After brief coverage on the program as a whole, the projects used in several courses are enumerated. This is followed by a discussion of the syllabus for the initial course, CS555 Secure Software Engineering.

II. GENERAL

When the program was launched the department had several professors with expertise in information security from its online masters in InfoSec and likewise several with software engineering expertise. However, only one existed (the author) whose expertise spanned the two areas. As a result, the first year of the program (2004-2005) contains only one course, the initial one (CS555), that fully integrated security and software engineering.

Only partially addressed during the programs first year, the full remedying of this problem began with a summer faculty seminar the author gave for six software engineering professors in the early summer of 2005. Readings included significant portions of Bishop's *Computer Security* [4] as well as a number of articles and reports including [35]. During that summer the author was also finishing the initial version of *Software Assurance: A Guide to the Common Body of Knowledge to Produce, Acquire, and Sustain Secure Software* [32] versions of which have subsequently underlain much of the instruction.

While the first two sets of graduates exited with a good understanding of secure software engineering and interest from employers was strong, the third set graduating in May 2008 are the first set to have a consistently strong integration of security and software engineering throughout their two years. This shows the time – two years – needed for professors to become fully familiar with material and courses to mature.

The second networking course includes an extensive, competitive attack and defend exercise in which each student has a computer to configure and defend and attacks other students' systems. Plans exist to include the software that students produce in the secure software engineering sequence into what they need defend on their machines.

III. PROJECTS

Both the initial software engineering course and the following three semester sequence are project based. The topics for first course's projects have included:

- System to supply parents photos of current, ongoing activity in their children's' daycare center over the Web
- Secure single person, single machine file system
- Game that avoids the problems described in *Exploiting Online Games: Cheating Massively Distributed Systems* by Greg Hoglund and Gary McGraw

The three semester sequence's projects include ones to produce

- Secure on-site storage
- Secure distributed Internet messaging (IM) for the financial community
- Secure off-site storage service

One notable feature involved in some was the allowing of students to assume the secure operation of the operating system and related services – except for those known to be ones to avoid. The alternative was for the students to use one of a few limited and unfamiliar OSs or to try (and presumably fail) to show the security of a familiar one.

These projects covered the development lifecycle and included a somewhat novel artifact/deliverable, the assurance case. An assurance case includes its top-level claim such as a safety or

[1] Also during the same period two undergraduate seminars on secure software engineering have been given using readings mainly from [32] and [31].

security-related claim, the arguments for this claim, and the evidence that supports these arguments.[2)] It is the central enabling mechanism for showing adequately low uncertainty, supporting relevant technical risk management, achieving grounds for appropriate confidence, and aiding in making related decisions.

IV. INITIAL COURSE CS555

The syllabus and related reading for the initial one semester course doing a first pass through the lifecycle are shown in Table 1. Generally, the students already have an undergraduate course in software engineering or relevant work experience, but almost never knowledge of software engineering security. Each row is a class in the course with 1.25 hours of classroom instruction. The level of participation of the instructor in project team meetings has varied from approximately 3 to 10 hours per team.

The readings are divided into two groups. While all have been used in one or more offerings in the course, they do not include a few used but found to be unsuitable. The groups reflect the requirements on the students, required (unlabelled) or "Optional". These also may reflect a degree of advice to those creating courses.

While the articles and reports are referenced by their author, year, and sometimes title, to aid brevity the author's last initial is used for the following books:

- B: *High-Assurance Design*, Clifford Berg, Addison Wesley, 2006
- G: *Building a Secure Computer System*, Morrie Gasser, Van Nostrand Reinhold, 1988
- R: *Software Assurance*, Samuel T. Redwine, Jr. (Editor), US Department of Homeland Security 2006
- S: *Software Engineering,8th Edition*, Ian Summerville, Addison Wesley, 2006

Table 1: CS555

Class	Topic	Read before Class (in order listed)
1	Introduction to Course, Software Engineering, Quality, Security	
2	Software Systems Engineering	S: Chapter 1.2, 2.1-2.2.2 R: 1-2 G: Chapters 1-2 Optional: S: Remainder of Chapters 1-2
3	Dependability, Security, Assurance and Assurance Case	G: 3 R: 3.0-3.3 Avizienis, Basic Concepts and Taxonomy of Dependable and Secure Computing Optional: Landwehr, 2001
		Optional: B: Chapter 1 CAA CAP670: Part B, Section 3[3] Preliminary Part
4	Security Principles, Critical Systems, Management Roles, Project Problem	R: 3.4-3.7 Redwine 2008 Sections 0 and 2 S: 3, 30.1
5	Projects and processes	S: Chapters 4, 20 NSA, IATF v3.1 Chapter 3, 2002 Redwine and Davis, 2004 Section 4 McGraw, Attacking Malicious Code, 2000
6	Requirements, Introduction to Security Functionality	S: Chapters 7, 9 G: Chapter 5 R: 4-5 Clark and Wilson 1987 Goodenough 2007 Presentation Optional: S: Chapters 6 Optional: B: Chapter 2
7	Project Management	S: Chapters 5 R: 11 S: 8.0-8.2 Redwine and Davis, Processes for Producing Secure Software 2004 Section 7
8	Introduction to Formal Methods	S: Section 10.1-2 R: 10-10.1 Hall and Chapman, Correctness by Construction 2002 Optional: S: 8.3-5
9	Configuration Management	S: Sections 29.0-29.3.1 S: Chapters 11
10	Architecture	S: Chapters 12 R: 6 S: 30.2-30.3, 32.0-32.1 B: Chapter 3 Redwine 2008 Section 2
11	Architecture, Planning	S: 12, 13 G: Chapters 4, 6, 9, 10, 11, and 13 Karger, VAX VMM 1990 Whitmore, Security Architecture 2001 B: Chapter 4 CAA CAP670: Part B, Section 3
12	Design, Team building	G: 12.0-12.5 S: Chapters 16 and 14

[2)] Or where appropriate in lieu of evidence, explicit assumptions

[3] Systems Engineering, SW 01 Regulatory Objectives for Software Safety Assurance in ATS Equipment

		Fernandez 2007 Optional: Irvine, Exemplar Project 2004 Optional: OpenGroup, Security Design Patterns S: 25
13	Assurance Case, Software Quality Assurance,	Optional: S: Chapters 26 S: Chapters 22 and 27 Redwine 2007
	Secure Software Assurance revisited, Software Construction	S: Chapters 24 and 20.2 Viega, Scanning Java 2000 SUN Java Coding Standards (Security) Look at: CWE and CAPEC websites Optional: Alexander, Coping with Java Stress, 2000
14	Software Construction	R: Section 7 S: Chapter 19
15	Project Assurance Cases: Presentations and Discussion	S: Chapter 22 Optional: Bishop, Software Lifecycle Security Checklist
16	Static Analysis, Inspections, Inspection Exercise; Due at end of class: Inspection report	S: Chapter 23 Look at: NIST SAMATE website
17	Student Status Presentations	
18	Testing	R: 8
19	VV& E	R: 9
20	Tools	S: 10.3
21	Z	S: Chapter 18.3-18.5, 19 Optional: S: Chapter 21
22	Evolution, Reuse	
22	Assurance Case, Using Cryptography	S: Chapter 2.4, 18.1-18.2, 30.4, 31.0-31.1, 17.0-17.2
23	Other Development Approaches , Professionalism, How to become world class	S: 1.2 Bratus 2007
24	Party, Demo, Present Lessons Learned	
25	Review	

In particular, the Sommerville textbook chapters on critical systems have been quite useful, and the eight edition's chapter on security also lends some help. One needs to keep alert for new literature appropriate for courses, and undoubtedly some will change for fall 2008. The readings listed here, however, have proven their worth – although readers should also make their own judgments particularly regarding optional material.

As discussed in the last section, the project constitutes a significant portion of student effort in addition to readings and class. Thus, the student load varies among students and totals 11-15 hours per week including class time.

Student reviews have tended to emphasize that they learned a lot in return for their, obviously, hard work. This initial course has been successful in providing a substantive initial background upon which students (and instructors) can build during the next three semesters.

V. CONCLUSION

This is the fourth year of the JMU graduate secure software engineering program. A considerable amount of experience has been acquired and lessons learned. Among these is a refinement of the initial course and its readings.

ACKNOWLEDGEMENTS

I want to mention a few of the key sources of help and ask forgiveness from those not listed. Professors Chris Fox and Ralph Grove have been instrumental in shaping the program. We have also been aided by Professors Steve Wang and Brett Tjaden. My colleagues on the DHS/DoD Software Assurance Working Groups and the somewhat related efforts in the IEEE, Object Management Group SIG on Software Assurance., National Defense Industry Association (US) Systems Assurance Committee, revision of ISO/IEC 15026, and elsewhere have contributing to my understanding including those involved in the 2006 IEEE Workshop on Secure Software Engineering Education and Training.

REFERENCES

[1] Alexander, Roger T., James M. Bieman, John Viega. Coping with Java Programming Stress. IEEE Computer, April 2000

[2] Anderson, Ross J. Security Engineering: A Guide to Building Dependable Distributed Systems. John Wiley and Sons, 2001.

[3] Avizienis, Algirdas, Jean-Claude Laprie, Brian Randell, and Carl Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pp. 11-33, Jan.-Mar. 2004.

[4] Bishop, Matt. Computer Security: Art and Practice, Addison-Wesley, 2003

[5] Bratus, Sergey, "What Hackers Learn that the Rest of Us Don't: Notes on Hacker Curriculum," IEEE Security and Privacy, vol. 5, no. 4, pp. 72-75, Jul/Aug, 2007

[6] CAA. CAP670 Air Traffic Services Safety Requirements. Civil Aviation Authority, UK, 2003

[7] CAPEC. Common Attack Pattern Enumeration and Classification at http://capec.mitre.org/

[8] Clark David D., and David R. Wilson. A Comparison of Commercial and Military Computer Security Policies. 1987 IEEE Symposium on Security and Privacy, IEEE, 1987

[9] CWE. Common Weaknesses Enumeration at http://cwe.mitre.org/:

[10] Dobbing, Brian, and Alan Burns. The Ravenscar Tasking Profile for High Integrity Real-Time Programs. SlGAda '98, 1 l/98 Washington, D.C., USA, ACM, 1998

[11] Fernandez, E. B., M. M. Larrondo-Petrie, T. Sorgente, and M. Vanhilst. A Methodology to Develop Secure Systems Using Patterns. 2007

[12] Gilliam, David P., Thomas L. Wolfe, Josef S. Sherif, Matt Bishop. Software Security Checklist for the Software Life Cycle. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), 2003

[13] Goodenough, John. Assurance Case, Presentation to DHS Software Assurance Working Group on Process and Practices, May 14, 2007

[14] Hall, Anthony, and Rodrick Chapman, Correctness by Construction: Developing a Commercial Secure System, IEEE Software, January/February 2002

[15] Irvine, Cynthia E., Timothy E. Levin, Thuy D. Nguyen, David Shi.ett, Jean Khosalim, Paul C. Clark, Albert Wong, Francis A.nidad, David

Bibighaus, Joseph Sears. Overview of a High Assurance Architecture for Distributed Multilevel Security, Proceedings of the 2004 IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY, 10–11 June 2004
[16] Irvine, Cynthia E., Timothy E. Levin, Thuy D. Nguyen, George W. Dinolt. The Trusted Computing Exemplar Project. Proceedings of the 2004 IEEE Workshop on Information Assurance and Security. United States Military Academy, West Point, NY, IEEE, 10–11 June 2004
[17] Irvine, Cynthia E., Timothy Levin, Jeffery D. Wilson, David Shifflett, Barbara Pereira, An Approach to Security Requirements Engineering for a High Assurance System. Naval Postgraduate School, 2002
[18] Karger, Paul A., Mary Ellen Zurko, Douglas W. Benin, Andrew H. Mason, and Clifford E. Kahn. A VMM Security Kernel for the VAX Architecture. 1990 IEEE Symposium on Security and Privacy, IEEE, 1990
[19] Kelly, T. "Tutorial: Software Safety Case Management," First OMG Software Assurance Workshop, March 2007.
[20] Kelly, T. and Weaver, R. "The Goal Structuring Notation – A Safety Argument Notation." Workshop on Assurance Cases: Best Practices, Possible Obstacles, and Future Opportunities, Florence, Italy. July 2004.
[21] Keqin Li, Laurent Mounier, Roland Groz, "Test Generation from Security Policies Specified in Or-BAC," compsac, pp. 255-260, 31st Annual International Computer Software and Applications Conference - Vol. 2- (COMPSAC 2007), 2007
[22] Landwehr, Carl, Computer Security, IJIS 1:3-13, 2001
[23] Lautieri, S., Cooper, D., and Jackson, D. "SafSec: Commonalities Between Safety and Security Assurance." Proceedings of the Thirteenth *Safety Critical Systems Symposium* - Southampton, 2005.
[24] Lesk, Michael, "The New Front Line: Estonia under Cyberassault," IEEE Security and Privacy, vol. 5, no. 4, pp. 76-79, Jul/Aug, 2007
[25] Leveson, Nancy. "A Systems-Theoretic Approach to Safety in Software-Intensive Systems," *IEEE Transactions on Dependable and Secure Computing* 1, 1 (January-March 2004): 66-86, 2004.
[26] Lockheed Martin. Resource Ordering and Status System (ROSS) Security Function Specifications. August 23, 1999
[27] McGraw, Gary, and Greg Morrisett. Attacking Malicious Code: A Report to the Infosec Research Council. IEEE Software, September/October 2000
[28] Ministry of Defence. *Interim Defence Standard 00-56, Safety Management Requirements for Defence Systems*, 17 December 2004.
[29] National Security Agency, The Information Systems Security Engineering Process (IATF) v3.1. 2002
[30] Open Group. Security Design Patterns (SDP) Technical Guide v.1, April 2004
[31] Payne, Charles N., Judith N. Froscher and Carl E. Landwehr. Toward A Comprehensive Infosec Certification Methodology. 16th National Computer Security Conference, Baltimore MD, Sept. 20 – 23, 1993, NCSC/NIST, pp. 165–172.
[32] Redwine, Samuel T,. Jr. Towards an Organization for Software System Security Principles and Guidelines, IIIA JMU, IIIA Pub. 0801, 2008
[33] Redwine, Samuel T., Jr. (Editor). Software Assurance: A Guide to the Common Body of Knowledge to Produce, Acquire, and Sustain Secure Software Version 1.1. US Department of Homeland Security, September 2006.
[34] Redwine, Samuel T., Jr. Dependability Properties, Technical Report, Commonwealth Information Security Center, James Madison University, 2004
[35] Redwine, Samuel T., Jr. The Quality of Assurance Cases, DNS Workshop on Assurance Cases for Security, 2007
[36] Redwine, Samuel T., Jr., and Noopur Davis (Eds.). Processes for Producing Secure Software. National Cyber Security Partnership, 2004
[37] SafSec Project. *SafSec Methodology: Guidance Material: Integration of Safety and Security.* Available at: http://www.praxis-his.com/safsec/safSecStandards.asp.
[38] SafSec Project. *SafSec Methodology: Standard: Integration of Safety and Security.* Available at: http://www.praxis-his.com/safsec/safSecStandards.asp.
[39] SAMATE. US NIST
[40] Salter, Chris, O. Sami Saydjari, Bruce Sehneier, Jim Wallner. Toward A Secure System Engineering Methodology. 1998 NSPW, Charlottesville, VA, USA, 1998
[41] Viega, John, and Gary McGraw, Tom Mutdosch, Edward W. Felten. Statically Scanning Java Code: Finding Security Vulnerabilities. IEEE SOFTWARE, September/October 2000
[42] Weaver, R. "The Safety of Software – Constructing and Assuring Arguments." Doctorial Thesis – University of York: Department of Computer Science. 2003.
[43] Weaver, R., Fenn, J., and Kelly, T. "A Pragmatic Approach to Reasoning about the Assurance of Safety Arguments." 8th Australian Workshop on Safety Critical Systems and Software (SCS'03), Canberra. 2003.
[44] Whitmore, J. J. A Method for Designing Secure Solutions. IBM Systems Journal, VOL 40, NO 3, 2001
[45] Williams, J. and Schaefer, M. "Pretty Good Assurance." Proceedings of the New Security Paradigms Workshop. IEEE Computer Society Press. 1995.