Information Security Awareness: A Course Module Using Simulated Spear-Phishing

Paula Crouse[1], Robert Farmer[2]
Mount Saint Vincent University
Canada
Paula.Crouse@msvu.ca
Robert.Farmer@msvu.ca

ABSTRACT: *With the introduction of information security awareness curriculum or modules in existing university courses comes the question of how to plan course content that will enhance students' learning. This paper outlines one active-learning approach—a simulated spear-phishing project that was utilized in information security awareness education at a post-secondary institution. The project framework is described, and insight is offered to assist information security educators to employ a similar project in their curriculum.*

Keywords: Information Security Awareness, Spear Phishing, Active Learning, Teaching Information Security

Received: 12 January 2016, Revised 18 February 2016, Accepted 3 March 2016

1. Introduction

Technology is everywhere, and the average user today owns more than one device—laptop, tablet, desktop and smartphone. As the number of devices increases and one's comfort level with each device improves, the same cannot be said for users' level of awareness about the risks associated with their use. Therefore, information security awareness education and training are timely and beneficial assets to formal education programs at universities and colleges (Ciampa, 2014; Mensch & Wilkie, 2011) as well as in the workplace (KnowBe4, 2015; PhishMe Company, 2015; ProofPoint, 2015). Drawing from several research studies, Grovo (2014), a provider of video training for Internet and modern professional skills, cites security and privacy as one of the eight digital skills essential for the 21st century workforce.

Almost daily, news reporters share accounts of security breaches that have occurred in small and large companies, international and local, high-profile and modestly publicized. Anyone could be affected by these breaches. Similarly, a leak in the system could occur through any employee at any level. As attackers favour the "human factor," traditional firewalls and mail filters, for example, do not sufficiently protect information resources (Ciampa, 2014; Eminaaolu, Uçar, & Eren, 2009; Fowler, 2011). Fraudsters continue to adapt their strategies to keep ahead of security defenses. While company executives were recently considered as the number one target of cybercriminals, ProofPoint, Inc. research reveals a shift in 2014 to tactics that focus on middlemanagement (2015). Since cybercriminals often rely on the weakest element—the user—(Bogdan, 2010; Fowler, 2011; Parmar,

2013; Young, 2008) it follows that employers in any field would benefit from new hires who bring a sense of information security savviness to their workplace.

One key social engineering approach to gaining access to personal or sensitive information comes through phishing. Phishing and its variations—for example, spear phishing—continue to be successful so these tactics are not expected to disappear soon. Research published by security software firm Trend Micro (2012) identified that 91 percent of cyberattacks had begun with a spear-phishing e-mail. Reports of successful spear-phishing attacks continue into 2015 (Kitten, 2015; Korolov, 2015).

As educators work with students to prepare them to be better equipped for the workplace and to more effectively utilize the technologies they have grown up with, courses in information security awareness or the addition of information security modules in other courses should be considered.

How does one approach effective learning activities in such a course? Active learning and project-based learning, which are shown to be effective in many disciplines (Ilvonen, 2013; Licht, 2014; Lucas, Testman, Hoyland, Kimble & Euler, 2013), can be used to facilitate learning where a student may be thinking that "this wouldn't happen to me." Information security textbook author Ciampa (2014) comments on the value of projects as they make learning "come alive" as students perform tasks.

This article describes the framework for a student project that was used in an information security awareness course offered as a special-topics class for senior students on a small university campus. Students, under the strict guidance of their professor, undertook a simulated spear-phishing project to educate themselves about phishing as well as their target audience—another student group with something in common.

As part of information security training in the workplace, companies such as PhishMe and KnowBe4 train employees to recognize phishing attempts and thereby reduce the risks associated with responding as the senders hope (KnowBe4, 2015; PhishMe Company, 2015). The student project described here is a variation of such workplace training, requires no budget, and can significantly enhance students' understanding about phishing and how easily one can fall victim.

2. Project Framework

2.1 Preparation of the Project—Professor Role

The overall project design was built by the course instructor in consultation with others on the university campus: another faculty colleague, the Departmental Research Ethics Board, Information Technology and Services Department, and the Housing Department. As with all student projects involving research with humans, it was important to secure ethics approval for this project. Given the proposed activity and the language in the Students' Computer Usage Agreement, it was also appropriate to discuss the project with senior staff in Information Technology and Services. While one goal was to ensure no negative repercussions of the simulated spear-phishing attack, it was also critical to keep staff informed in case the "victims" approached the Information Technology and Services Help Desk for assistance.

The Housing Department manager was selected as an ideal individual to "get on board" as she could provide the avenue to potential phishing victims. To make the spear-phishing attack appear real, a group with something in common was necessary as the target audience. It was agreed that the students in residence would serve as the targets. The number of residence students would provide a reasonable size group for the study, and the e-mail addresses of this group were readily available without extracting them from a larger database. Only university-assigned email addresses were used by the "phishers"; however, students may have had their university address forwarded to a personal e-mail address. As the project scope was being established, detailed information about the project framework and goals was shared, and the Housing Department manager was invited to probe and express any concerns. If residence students later raised questions or concerns, it was critical that she and her staff were prepared to address them yet had confidence in contacting the course instructor, if needed.

The course instructor prepared a confidentiality agreement for each student enrolled in the information security class to sign. If the students happened to access sensitive information about students, employees, or the University during the course of this project, each student agreed that the information would be held in strict confidence and that the project specifics would be shared only for the purpose of fulfilling the course requirements.

The final preparation for the project was the selection of an e-mail program to be used to carry out the simulated phishing attack.

A program was selected that met these criteria: (a) permitted a login using an e-mail account created for the purpose of this project, (b) showed links as text, not URLs, (c) supported a blind copy field so that student addresses would not be visible to other students, (d) permitted multiple recipient addresses without spam filters potentially flagging the messages as risky, and (e) could be installed easily and quickly on the computer to be used for sending messages.

2.2 Preparation for Simulated Spear-Phishing—Student Role

Following class introduction to the project and a general discussion of phishing and its variants, students conducted a literature review in two areas. First of all, they learned about spear-phishing and the common characteristics of spear-phishing messages that were presently being distributed. They also researched how prevalent spear-phishing attempts were and learned about "success" cases. Secondly, students researched the role of controlled fake spear-phishing attacks in a business' ongoing IT security education plan.

Following their research, the students chose a message topic that would be relevant to the target audience—the students living in residence. Since all students must purchase some form of meal plan, they began to prepare their spear-phishing message about a malfunction in the system holding meal plan records. As is typical of spear-phishing attempts, the tone of the message depicted urgency and encouraged the intended victims to respond immediately by clicking on one of the links in the message. To collaborate on this message content, students wrote the message using a wiki.

The students in the course also collaborated in a wiki to write educational content to share with their peers living in residence, if the recipient of the message did follow through to click on a link. All links, whether to seek more information or to find out if one's own meal plan account had been compromised, took the user to this educational website. To begin with, the "victim" was immediately assured that no meal plan records had been compromised and that the residence student had unknowingly been part of a course project. The website also informed the "victim" that no personal information had been revealed to the students in the class. The website briefly outlined the purpose of the project and then shared five tips for recognizing e-mail scams.

All written content was submitted to the course instructor for careful review prior to its being approved for use.

2.3 Executing the Spear-Phishing Attempt—Professor Role

Once the spear-phishing message and the website content were approved, the educational website content was uploaded. Following testing, the course instructor and another colleague visited the Housing Department and used a computer in that department to send out the spear-phishing messages. A member of the Housing Department was present during this time. The e-mail program chosen for this study was installed on this computer, and the e-mail addresses of the students in residence were opened in a spreadsheet file.

In addition to copying the message content, the student addresses in groups of ten were copied and pasted into the blind copy field. A copy of all sent messages was retained to verify the quantity of messages that were sent. Then, prior to the professors' leaving the Housing Department, those sent items were deleted to ensure the privacy of the students in residence, and the e-mail program was uninstalled.

2.4 Website Monitoring—Professor Role

The educational website was monitored by the professors for one week, and the website statistics were shared with the students in the course. Statistics included the number of unique visits to the site, the date, and the operating system from which the visitor connected. With the analytics available for this website, it was not possible to identify any of the visitors. In this project, it was important to maintain anonymity so that anyone who fell victim to the phishing attempt would not feel embarrassed.

2.5 Research Paper—Student Role

To conclude this project, students wrote a group paper and submitted it for assessment. The paper summarized the literature review conducted earlier, explained the methodology used for the study, recorded the results of the website monitoring, and analyzed and discussed the results.

3. Conclusion and Insight

The project was a valuable learning opportunity for the students in this information security awareness course, and by their participation in the project, they felt they were so much better prepared to ward off attacks than if they had not had this research

experience. They learned how easy it is for anyone to be fooled by phishing ploys and were astounded by the number of unique visitors to their website within the first 24 hours of the message having been sent. With more time in the semester, it would have been interesting to provide an opportunity for those who had been "phished" to provide feedback to the researchers.

Beyond the primary goal of this project—greater information security awareness—many other learning opportunities can be realized. Such outcomes may include improved collaborative writing skills, better communication with group members, development of research skills, proactive thinking, and familiarity with new technology skills such as work with a wiki.

To replicate this project in your information security awareness course or module, plan the project details in advance and get the key people "on side." Not everyone will embrace such a project, nor will they understand it. Prior to beginning this project, the instructors were very aware of the prevalence of phishing attempts to student e-mail addresses as well as anecdotal information about their success rate. In addition to meeting the course goals, a secondary goal was to play one small part in reducing the number of students who were responding to such phishing attempts without questioning their legitimacy. This type of student project has the added value of benefitting people outside of the classroom.

Know the ethics standards within such a project must operate. For example, students must not craft messages that appear to come from an external, legitimate company and perhaps abuse trademarks or risk jeopardizing a company's reputation. Consider the ramifications of the chosen scenario the students create.

Know in advance how you wish to monitor the website hits to ensure that the analytics software is in place. For example, special features may be necessary if it were desirable to monitor hits from mobile devices versus desktops.

Communicate with all potentially involved parties at all stages of the project. Communicate with student researchers. For much of this project, the educator role becomes one of facilitator while the students take on an active-learning role. This does not, however, equate to the educator's being disconnected from the phases of the project. Be prepared to provide guidance when needed.

4. Project Limitations and Generalizability

This simulated spear-phishing project is intended as an active-learning opportunity in a course and lacks the follow-up training and repetition that should be inherent in workplace phishing awareness training (KnowBe4 Visible Proof, 2015; Lindros & Tittel, 2014; Parmar, 2012; PRWeb, 2014). The length of the academic semester does not afford the same opportunities as established businesses enjoy. For example, it is hoped that the students in residence who received the targeted spear-phishing message and subsequently visited the website now have a greater awareness of e-mail risks and will seek to learn more about protecting their personal information as well as that of their employers. In the workplace, however, some form of training can follow the phishing e-mail and later simulated phishing attacks can be monitored to determine the effectiveness of the training. Done right, educational campaigns to reduce the effects of spear-phishing are effective (Eminaaolu, Uçar, & Eren, 2009; Osterman Research, 2015; PhishMe ROI, 2015).

References

[1] Bogdan (2010, February 25). Computer security demands tech savvy employees [Alertsec information security blog]. Retrieved from http://blog.alertsec.com/2010/02/computer-security-demands-tech-savvy-users/.

[2] Ciampa, M. (2014), Security Awareness: Applying Practical Security in Your World (4th ed). Boston, Massachusetts: Course Technology.

[3] Eminaaolu, M., Uçar, E., Eren, ^. (2009). The positive outcomes of information security awareness training in companies—A case study. Information Security Technical Report, I14, pp. 223-229.

[4] Fowler, G. (2011, September 11). What's a company's biggest security risk? You. The Wall Street Journal.

[5] Grovo (2014). How the digital skills gap is killing productivity and what you can do about it. Retrieved July 17, 2014, from http://whitepapers.lakewoodmediagroup.net/content/how-digital-skills-gap-killing-productivity-and-what-you-can-do-about-it.

[6] Ilvonen, I. (2013). Information security assessment of SMEs as coursework—Learning information security management by

doing. Journal of Information Systems Education, 24 (1).

[7] Kitten, T. (2015, January 2). Spear-Phishing: A bigger concern in 2015. Bank Info Security. Retrieved from http://www.bankinfosecurity.com/spear-phishing-bigger-concern-in-2015-a-7742/op-1.

[8] KnowBe4 (2015). www.knowbe4.com/about-us/.

[9] KnowBe4 (2015). Visible Proof the KnowBe4 System Works. Retrieved April 8, 2015, from http://www.knowbe4.com/visible-proof-the-knowbe4-system-works/.

[10] Korolov, M. (2015, February 13). Omaha's Scoular Co. loses $17 million after spearphishing attack. CSO. Retrieved from http://www.csoonline.com/article/2884339/malware-cybercrime/omahas-scoular-co-loses-17-million-after-spearphishing-attack.html.

[11] Licht, M. (2014, October). Controlled chaos: Project-based learning. The Education Digest, pp. 49-51.

[12] Lindros, K. and Tittel, E. (2014). How to test the security savvy of your staff. CIO. Retrieved from http://www.cio.com/article/2378559/data-breach/how-to-test-the-security-savvy-of-your-staff.html.

[13] Lucas, D., Testman, J., Hoyland, M., Kimble, A., and Euler, M. (2013). Correlation between active-learning coursework and student retention of core content during advanced pharmacy practice experiences. American Journal of Pharmaceutical Education, 77 (8).

[14] Mensch, S. and Wilkie, L. (2011). Information security activities of college students: An exploratory study. Academy of Information and Management Sciences Journal, 14 (2), 91-115.

[15] Osterman Research, Inc. (2015). Best practices for dealing with phishing and next-generation malware. Retrieved April 8, 2015 from http://info.knowbe4.com/whitepaper-osterman-bp-phishing.

[16] Parmar, B. (2012, January). Protecting against spear-phishing. Computer Fraud & Security, pp. 8-11.

[17] Parmar, B. (2013, March). Employee negligence: the most overlooked vulnerability. Computer Fraud & Security, pp. 18-20.

[18] PhishMe (2015). Company Overview. http://phishme.com/company/overview/.

[19] PhishMe (2015). Return on Investment. Retrieved April 8, 2015, from http://phishme.com/the-phishme-advantage/roi/.

[20] ProofPoint, Inc. (2015). The human factor 2015. Retrieved April 29, 2015, from https://www.proofpoint.com/us/id/WP-Human-Factor-Report-1.

[21] TrendMicro (2012). Spear-phishing e-mail: Most favored APT attack bait. Retrieved April 8, 2015, from http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-spear-phishing-email-most-favored-apt-attack-bait.pdf.

[22] PRWeb (2014). Key findings from security awareness training survey unveiled by Security Monitor and Enterprise Management Associates [Online press release distribution service]. Retrieved April 24, 2015, from http://www.prweb.com/releases/survey-results-security/awareness-training/prweb4337664.htm.

[23] Young, J. R. (2008, December 19). Top 10 threats to computer systems include professors and students. Chronicle of Higher Education, 55 (17), p. A9.

1. Introduction

In the past decade, software security has become an increasingly important area of interest due to the fact that the majority of information security vulnerabilities are due to software defects. There have been a number of initiatives to integrate security into software or product development lifecycle (ENISA 2011). Some of these initiatives include Microsoft's trustworthy computing initiative, the Building Security in Maturity Model (BSIMM) (BSIMM 2014), the Software Assurance Maturity Model (SAMM) (OPENSAMM 2014), etc. As a result, academic community began to realize the importance of integrating software security education into computing curricula in universities and colleges.

The Department of Homeland Security (DHS) Software Assurance (SwA) program developed the secure software assurance Common Body of Knowledge (SwA 2007) which can be used to guide the development of education and training curricula related to software assurance. Sponsored by DHS, faculty and researchers from Software Engineering Institute (SEI) at Carnegie Mellon University (CMU) and other institutes developed a Master of Software Assurance Reference Curriculum. The new ACM/IEEE draft Computer Curricula 2013 (ACM/IEEE 2013) also includes secure software engineering knowledge units.

Some universities have initiated the implementation of secure software engineering education through developing graduate or undergraduate programs in software assurance, developing courses on software security, or course modules which can be integrated into typical computer science courses. There have been also hands-on labs developed which are suitable for teaching software security. Industry and open source community initiatives have contributed abundant resources that educators can draw on to develop secure software engineering curricula.

This paper provides an overview of current efforts and resources in secure software engineering education, with the goal of providing guidance for educators to make use of these resources in developing secure software engineering curriculum. These resources include Common Body of Knowledge, reference curriculum, sample curriculum materials, hands-on exercises, and resources developed by industry and open source community.

We investigate the relationship among the Software Assurance Knowledge Areas/Common Body of Knowledge by DHS, the core body of knowledge presented in the reference curricula proposed by SEI at CMU, and the knowledge areas presented in ACM/IEEE through mapping or cross-referencing them. The recent practices on secure software engineering education, including secure software engineering related programs, courses, and course modules are discussed. The course modules are categorized into four categories to facilitate adoption. Available hands-on exercises developed for teaching software security are described and mapped to the taxonomy of coding errors proposed by McGraw (2006). The rich resources including various secure software development processes, methods and tools developed by industry and open source community are surveyed. A road map is provided to organize these resources and guide educators in adopting these resources and integrating them into their courses.

The remainder of the paper is structured as follows. Section 2 presents the knowledge areas of secure software engineering defined by DHS, CMU and ACM/IEEE, and the mapping among them. Section 3 describes the practices and implementation of secure software engineering education, including software security programs, concentrated courses on secure software engineering, teaching modules on secure software engineering, and hands-on exercises for teaching software security. Section 4 presents resources provided by industry and open source community, and a roadmap to these resources. Section 5 concludes the paper.

2. The Secure Software Assurance Common Body of Knowledge

Knowledge areas of secure software assurance as an important education component are defined by both government agencies such as DHS and academic organizations such as CMU, ACM and IEEE. Knowledge areas defined by these organizations can be mapped and related to each other.

2.1 Secure Software Assurance Common Body of Knowledge from DHS

The Department of Homeland Security Software Assurance (SwA) Program developed the Secure Software Assurance Common Body of Knowledge (SwA CBK) (SwA 2007) with the goals of identifying workforce needs for competencies, representing best practices, and providing guidance for developing education and training curricula related to software assurance. The topics included in the SwACBK are summarized in Table 1.

Knowledge Area	Topics
Dangers and Damages	attackers, attacks, known vulnerabilities and exploits
Fundamental Concepts and Principles	dependability, security, assurance, basic software system security principle, safety, secure software engineering
Ethics, Law, and Governance	ethics, laws, regulations, and standards peculiar to developing secure software
Secure Software Requirements	security needs, requirements analysis, specification(assumption, security policy), security functional requirements, requirement validation, assurance case
Secure Software Design	design objectives, principles and guidelines for designing secure software, architectures for security, security functionality, proper use of encryption and encryption protocols, frameworks, design patterns for secure software, database security, methods for tolerance and recovery, deception and diversion, software protection, forensics support, user interface design, assurance case for design, secure design process and methods, design reviews for security
Secure Software Construction	common vulnerabilities, construction of code, secure coding, construction of user aids, secure release
Secure Software Verification, Validation and Evaluation	assurance case and tools, ensuring proper version, testing process and techniques, dynamic and static analysis, usability analysis, verification and validation of user aids, secure software measurement
Secure Software Tool and Method	formal and semi-formal methods, compilers, static and dynamic analysis, development tool suites, selecting tools
Secure Software Processes	heavyweight and lightweight processes, legacy upgrade processes, security of developmental process, improving processes for developing secure software
Secure Software Project Management	project risk management, selecting a secure software process, security management, assuring security level of software shipped, secure configuration management, software quality assurance and security
Secure Software Sustainment	operational assurance, response management, infrastructure assurance

Table 1. Knowledge Areas and Topics Defined by DHS

2.2 Software Assurance (SwA) Reference Curriculum from CMU

The Software Assurance (SwA) Reference Curriculum from CMU (Mead, Allen, Ardis, Hilburn, Kornecki, Linger & McDonald 2010a) identifies and presents a core body of knowledge from which to create a Master of Software Assurance standalone degree program, or a track within existing software engineering and computer science program. The outcomes graduates can expect after completing this program are organized into Assurance Process Management (APM) and Assurance Product and Technology (APT). The APM consists of assurance across life cycles, risk management, assurance assessment, and assurance management. The APT includes system security assurance, system functionality assurance, and system operational assurance.

Each outcome was captured as a knowledge area for the core body of knowledge (BOK), each knowledge area includes a set of knowledge units with assigned cognitive levels based on the Bloom's Taxonomy. The curriculum also describes student prerequisites, and provides recommendation for faculty considering implementing such a program.

The Master of Software Assurance (MSwA) Reference Curriculum was recognized by the IEEE Computer Society (IEEE-CS) and ACM as appropriate for a master's program in software assurance in 2010 (IEEE 2010). The Master of Software Assurance

Course Syllabi provides sample syllabi (Mead, Allen, Ardis, Hilburn, Kornecki, Linger & McDonald 2011a) for nine (9) core courses in the Master of Software Assurance Reference Curriculum. The nine (9) core courses are:

1) Assurance Management Course: It covers risk management for assurance; compliance with laws, regulations, standards, and policies related to assurance; assurance practices in planning and managing development projects; and making the business case for assurance.

2) System Operational Assurance Course: It discusses how to establish procedures to assure that working systems continue to meet their security requirements and can provide a response to new threats.

3) Assured Software Analytics Course: It covers analysis methods, techniques, and tools to help assure that newly developed and acquired software, systems, and services meet their functional and security requirements.

4) Assured Software Development 1 Course: It introduces the fundamentals of incorporating assurance practices, methods, and technologies into software development and acquisition life-cycle processes and models.

5) Assured Software Development 2 Course: It covers rigorous methods for specifying assurance requirements and for architecting and designing software and systems to meet those requirements.

6) Assured Software Development 3 Course: It discusses methods, techniques, and tools for developing secure code.

7) Assurance Assessment Course: It discusses fundamentals of establishing a required level of software and system assurance. Topics include assessment methods; product and process measures and other performance indicators; measurement processes and framework; and performance indicators for business survivability and continuity.

8) System Security Assurance: It discusses how to incorporate effective security technologies and methods into new and existing systems.

9) Software Assurance Capstone Experience: In this course students will develop or modify a significant software system using software assurance knowledge learned from courses throughout the program.

2.3 Secure Software Engineering Components in ACM/IEEE 2013

ACM/IEEE draft Computer Science Curricula 2013 (ACM/IEEE 2013) includes secure software engineering related topics in Knowledge Areas such as Information Assurance and Security (IAS), Programming Languages (PL) and Software Engineering (SE). These topics are listed in Table 2.

Knowledge Area	Tier	Topics
IAS/Principles of Secure Design	1	least privilege and isolation ,fail-safe defaults, open design, end-to-end security, defense in depth, security by design, tensions between security and other design goals
IAS/Principles of Secure Design	2	complete mediation, use of vetted security components, economy of mechanism (reducing trusted computing base, minimize attack surface), usable security, security composability, prevention, detection, and deterrence
IAS/Defensive Programming	1	input validation and data sanitization, choice of programming language and type-safe languages, examples of input validation and data sanitization errors (buffer overflows, integer errors), race condition, correct handling of exceptions and unexpected behaviors
IAS/Defensive Programming	2	correctly generating randomness for security purposes, correct usage of third-party components, security updates

IAS/Defensive Programming	Elective	information flow control, mechanisms for detecting and mitigating input and data sanitization errors, fuzzing, static analysis and dynamic analysis, program verification, operating system support (e.g., address space randomization, canaries), hardware support (e.g., DEP, TPM)
IAS/Web Security	Elective	web security model; session management, authentication; application vulnerabilities and defenses client-side security; server-side security tools
IAS/Secure Software Engineering	Elective	building security into the Software Development Lifecycle; secure design principles and patterns; secure software specification and requirements; secure coding techniques to minimize vulnerabilities in code, such as data validation, memory handling; crypto implementation; secure testing (including static and dynamic analysis); software quality assurance and benchmarking measurements
PL/Object-Oriented Programming	1	definition of classes: fields, methods, and constructors
PL/Object-Oriented Programming	2	object-oriented idioms for encapsulation (privacy and visibility of class members)
SE/Software Construction	2	defensive coding practices; secure coding practices
SE/Software Construction	Elective	potential security problems in programs (buffer and other types of overflows; checking input)

Table 2. Knowledge Areas Related to Secure Software Engineering in ACM/IEEE 2013

2.4 The Relationship among the Software Assurance Knowledge Areas from DHS, CMU and ACM/IEEE 2013

Software Assurance CBK from DHS	**MSwA BOK from CMU**	**ACM/IEEE 2013 Curricula**
Dangers and Damages	System Security AssuranceRisk Management	IAS/Threats and Attacks
Fundamental Concepts and Principles	System Security Assurance	IAS/Foundational Concepts in Security
Ethics, Law, and Governance	Assurance Management	IAS/Security Policy and Governance
Secure Software Requirements	Assurance Across Life Cycles	IAS/Secure Software Engineering
Secure Software Design	Assurance Across Life Cycles	IAS/Principles of Secure DesignIAS/ Secure Software Engineering

Secure Software Construction	Assurance Across Life Cycles	IAS/Defensive Programming SE/Software ConstructionPL/Object Oriented Programming
Secure Software Verification, Validation and Evaluation	Assurance Across Life Cycles System Functionality Assurance	IAS/Secure Software Engineering
Secure Software Tool and Method	System Security Assurance	IAS/Secure Software Engineering
Secure Software Processes	Assurance Across Life Cycles	IAS/Secure Software Engineering
Secure Software Project Management	Risk ManagementAssurance Management	IAS/Secure Software Engineering
Secure Software Sustainment	Assurance AssessmentAssurance ManagementSystem Operational Assurance	

Table 3. Mapping of Common Body of Knowledge from DHS, CMU and ACM/IEEE 2013

From Table 3, we can see the knowledge areas related to secure software engineering in ACM/IEEE 2013 covers the SwA CBK from DHS well. Both the SwA CBK and the knowledge areas related to secure software engineering in ACM/IEEE 2013 are specific. The BOK from CMU are outcome of nine courses recommended by CMU, which provides a sample curriculum for a Master of Software Assurance. Since the MSwA BOK from CMU is for graduate level study, they cover more advanced topics. The MSwA BOK from CMU can still cross-reference to DHS as shown in Table 3.

3. Practices And Implementation of Secure Software Engineering Education

Based on the resources a university has, there are three approaches to implement secure software engineering curricula:

1) Concentrated courses. One or several semester-long courses focusing on secure software engineering, software assurance, or secure coding are developed and taught.

2) Diffusion through curricula. This approach is to develop course modules and integrate them into existing courses in computer science or other related disciplines.

3) Concentration/degree program. This is to develop a concentration or track at the undergraduate or graduate level, or develop a standalone software assurance degree program. This approach may include the integration of the concentrated courses approach and the diffusion throughout curricula approach.

The advantages and disadvantages of the three approaches are listed in Table 4 (Chu, Stranthan, Cody, Peterson, Wenner & Yu 2009).

In what follows, we describe some of the example programs on secure software engineering, courses, and course modules.

3.1 Software Security Programs

Software security programs can be offered at graduate levels such as a Master degree program, a graduate certificate, or a track.

3.1.1 Graduate programs of secure software engineering

Stevens Institute of Technology offers a Masters Degree in Software Engineering with a Concentration in Software Assurance (Stevens 2014). This concentration program has two tracks: (1) Developing Trusted Systems and (2) Managing Trusted Systems. Both tracks require the same six core courses. Each track requires four additional courses special for the track. Stevens Institute of Technology also offers two Graduate Certificates: (1) Development of Trusted Software Systems; (2) Acquisition and Management of Trusted Software Systems. Each certificate requires four courses. These curricula are based on the Software Assurance Curriculum developed by CMU sponsored by DHS (Mead et al. 2010a).

Approach	Advantage	Disadvantage
Concentration/degree/certificate program	• Students can specialize in the secure software engineering field	• Need resources and institutional support
Concentrated courses	•Can introduce the subject quickly • Can cover specialized and advanced topics	• Students may not take these courses if they were elective courses· •There may not be enough room in the program of study to add these courses • Not able to reinforce secure software engineering techniques throughout the curriculum· • The department or program may not be able to offer these new courses due to shortage of instructors
Diffusion through curricula	• Can reinforce secure software engineering techniques throughout the curriculum • Do not add extra pressure on already-overburdened undergraduate degree programs	• There may not be enough class time to cover both the course topics and secure software engineering issues • It is challenging to train a large number of faculty members in secure software engineering· • It is unrealistic to expect faculty who teach subject matters to be up to date on latest attack vectors

Table 4. Approaches to Implement Secure Software Engineering Curricula

Northern Kentucky University (NKU) offers a graduate certificate in the field of secure software engineering. This graduate certificate program requires students to take four courses, which are Computer Security (CSC 582), Advanced Programming Workshop (CSC 601), Advanced Software Engineering (CSC 640), and Secure Software Engineering (CSC 666) (NKU 2014).

The Department of Computer Science at North Carolina Agricultural and Technical State University (NC A&T) developed a Secure Software Engineering Track in its graduate program (Yuan, Hernandex, Wadell, Chu & Yu 2012b). This track requires four courses including Software Specification, Analysis & Design (COMP710), Secure Software Engineering (COMP727), Software Security Testing (COMP725), and an elective course in software engineering or information assurance (Yuan et al. 2012b).

3.1.2 Undergraduate curriculum specialization for software assurance

The proposed undergraduate curriculum specialization for software assurance by CMU (Mead, Hilburn & Linger 2010b) includes a group of 7 courses: Computer Science 1 (CS 1), Computer Science (CS 2), Introduction to Computer Security, Software Security Engineering, Software Quality Assurance, Software Assurance Analytics and Software Assurance Capstone Project. The course description, prerequisites, syllabus, sources, course delivery features, and course assessment features, are described. The CS

1 and CS 2 courses are traditional CS 1 and CS 2 courses based on Computing Curriculum 2001 with modest changes to emphasize certain software assurance fundamentals (for example, secure coding).

3.1.3 Community college education on software assurance

The Software Assurance Curriculum Project Volume 4 from CMU (Mead, Hawthorne & Ardis 2011b) outlines community college courses for Information Assurance. The intention of the courses outlined in this document is to provide students with fundamental skills for continuing with undergraduate education or to provide those students with prior undergraduate technical degrees with additional knowledge to become more specialized in software assurance. The course outline and outcomes were developed based on the Guidelines for Associate Degree Transfer Curriculum in Computer Science (ACM 2009) and the proposed IA body of knowledge in the ITiCSE workshop report (Cooper, Nickell, Pérez, Oldfield, Brynielsson, Gökce, Hawthorne, Klee, Lawrence & Wetzel 2010). Six (6) courses were described in the report and they are: Computer Science 1, Computer Science 2, Computer Science 3, Introduction to Computer Security, Secure Coding, and Introduction to Assured Software Engineering.

3.2 Concentrated Courses on Secure Software Engineering

Software security can be taught in concentrated courses such as secure programming, vulnerability assessment or software security testing, and secure software engineering.

A course on "*Secure Programming"* (CS390S) was offered in the computer science department at Purdue University (Purdue 2014). This course covers low-level mistakes, as well as secure programming principles and ideas. Topics include Shell and environment, buffer overflows, integer overflows, format strings, meta-character vulnerabilities and input validation, web application issues, race conditions, file system issues, and randomness.

A course on "*Software Vulnerability Assessment*" was offered by the Laboratory of Information Integration Security and Privacy at University of North Carolina at Charlotte, in which secure software development is taught (Chu et al. 2009). The course trains students to have hacking mentality. Students learn various types of vulnerabilities, how to exploit them, and how to properly fix these security flaws through secure software design and implementation. The Department of Computer Science at North carolina A&T State University offers a course on "*Software Security Testing*" which focuses on software security testing techniques and tools (Yuan et al. 2012b). It covers various design and implementation vulnerabilities and how to prevent them, creating test plans based on risk analysis, black-box, white-box and gray-box security testing, fault injection etc. Security testing tools are used to test web applications.

A seminar course on "*Secure Software Engineering*" was taught by Walden and Frank (2006) at Northern Kentucky University. The course included a set of secure software engineering teaching modules such as what is software security; threats and vulnerabilities; risk management; security requirements; secure design principles and patterns; data validation; using cryptography securely; code review and static analysis; security testing; create a software security program. In the course, the students will work in a team to develop a secure web application by applying what they learn from these modules. Yuan et al. (2012b) developed a course on "*Secure Software Engineering*" in the Department of Computer Science at North carolina A&T State University. The course discusses how to incorporate security throughout the software development lifecycle. The main topics include security problems in software and methodologies to solve the problems, risk management framework for softwaresecurity, the software security touchpoints as best practices, code review and tools, architectural risk analysis, risk-based security testing and software penetration testing approaches, abuse cases, and categories of coding errors.

3.3 Teaching Modules on Secure Software Engineering

Teaching modules are useful in incorporating secure software engineering knowledge into existing curriculum. A collection of teaching modules to improve student learning on software security have been developed. The Software Engineering Institute at CMU provides lecture materials and artifacts that faculty can use to thread software assurance knowledge in their curricula (SEI 2014a). The Denim Group has developed e-Learning materials to expose students to the concepts of defensive programming (ThreadStrong 2014). Those materials include features such as detailed examples, multi-media and text lessons, interactive quizzes with review questions, audio instruction, etc. The Security Injection Project at Towson University (Towson 2014) developed a number of security injection modules which can be integrated into such courses as Computer Literacy, CS 0, CS I, CS II, and other courses such as Web Development, Database, and Networking. Each teaching module includes a background description and laboratory assignments. The background part includes a short description of the module topic, the type of risk involved, a real world example, and some exercises. The laboratory assignments provide engaging learning experience for students. The Department of Computer Science at NC A&T developed a series of course modules and integrated them into

undergraduate courses (Yuan 2012b).

We organize these teaching modules into three categories: 1) secure coding, 2) software threat and attack analysis, 3) secure application development, and 4) Software Security and Secure Software Engineering. Each category includes several topics. For each topic, there are one or more modules from different sources. Table 5 lists these teaching modules using this organization.

Module Category	Applicable Courses	Module Topics and Sources
Secure Coding	CS1, CS2, Data Structure, Algorithm, Programming languages	Secure coding (ThreadStrong 2014; Yuan et al. 2012b)Secure coding for .Net (ThreadStrong 2014) Secure programming (SEI 2014a;Towson 2014; Yuan et al.2012b) Secure data structure and algorithms (Yuan et al. 2012b)
Software Threat and Attack Analysis	Software Engineering, Networking, Web Application, Database Systems	Threat modeling (ThreadStrong 2014) Cross-site request forgery (Thread Strong 2014)Insider threat (SEI Database 2014a) security (Towson 2014;Yuan et al. 2012b)
Secure Application Development	Software Engineering, Web Programming, Mobile Computing	Overview of mobile application security (ThreadStrong 2014) Authentication and authorization of iOS/Android mobile devices (ThreadStrong 2014)Web application security (Thread Strong 2014)
Software Security and Secure Software Engineering	Software Engineering, Capstone Project, Computer Program Design	Security quality requirement engineering (SEI 2014a)Software quality analysis (SEI 2014a)Secure software engineering (SEI 2014a; Yuan et al. 2012b)Formal specificationand verification (SEI 2014a)Software security remediation (ThreadStrong 2014)

Table 5. Teaching Modules on Secure Software Engineering

3.4 Hands-on Exercises for Teaching Software Security

The SEED project (Syracuse 2014) developed a series of lab exercises for computer security education. Some of these labdemonstrate common vulnerabilities and attacks such as buffer overflow vulnerability, format string attack, Cross Site Scripting Attack, SQL Injection Attack, Click Jacking Attack, etc. Some of these lab exercises provide students with opportunities to apply security principles in designing and implementing systems, such as implementing firewall, access control mechanism, encryption, and sandbox, etc. Some of these lab exercises allow students to apply security principles in analyzing and evaluating systems, such as exploring Linux firewall, packet sniffing and spoofing, access control in Linux, etc.

OWASP WebGoat (OWASP 2014a) is a J2EE web application that was designed to be intentionally insecure in order to teach web application security lessons. Each lesson requires the user to demonstrate their understanding of vulnerabilities by exploiting security issues that are present in the application. It also provides hints and code that gives explanations of each lesson in further detail.

The SWEET (Secure WEb dEvelopment Teaching) project (Chen, Tao, Li & Lin 2010; Pace 2014) developed a set of portable teaching modules for secure web development. Some of these modules include software security related topics such as “Introduction to Cryptography”, “Secure Web Transactions”, “Web Application Threat Assessment”, “Web Server Security Testing”, “Java Security”, etc. Each module includes an introduction of the fundamental concepts, and lab exercises on these topics.

Kingdoms	Phylum	SEED Labs	Security Injection	WebGoat
Input validation	Buffer Overflow	Buffer-Overflow Vulnerability Lab	Input Validation Module Buffer Overflow Module	Off-by-One Overflow
	Command Injection			Command Injection
	Cross-Site Scripting	Cross-Site Scripting Attack Lab		LAB: Cross Site Scripting
	Format String	Format String Vulnerability Lab		
	Integer Overflow		Integer Errors Module	
	SQL Injection	SQL Injection Attack Lab		LAB: SQL Injection
API Abuse	Directory Restriction	Chroot Sandbox Vulnerability Lab		
	Often Mis-used: Privilege Management	Set-UID Program Vulnerability Lab		Using An Access Control MatrixBypassa Path Based Access Control
Security Features	Missing Access	ClickJacking Attack Forgery Attack Lab		Lab Cross-Site Request Control
Time and State	File Access Race Condition: TOCTOU	Race-Condition Vulnerability Lab		LAB: Role Based Access Control
Error Handling				Fail Open Authentication Scheme

Table 6. The Mapping of Hands-on Labs to the Taxonomy of Coding Errors 1

Chu et al. (2009) developed a set of hands-on exercises that teach vulnerability assessment alongside secure software development concepts. A number of example applications such as "Bog: Blog Server", "Tunestore: Online Music Store", "Tickle~Mail:Web-based Email", "Tickle~Shop: E-Commerce Site", "NCCure Health Care Systems: HR Benefit Management", and "Security First: Online Banking" are used to teach vulnerabilities due to data validation, logic error and design flaws. The vulnerabilities these exercises demonstrate include SQL injection, stored and reflective cross site scripting, DOM attack, email injection, poor design of password functions, privilege escalation, etc.

Yuan et. al (2012a) described some lab exercises that cover vulnerability assessment, static analysis with Fortify, fuzz testing with WebScarab, threat analysis and modeling, etc. Williams (2013) developed a lab exercise demonstrating stack overflow attack.

McGraw (2006) describes a taxonomy of coding errors that is composed of two distinct kinds of sets: phylum and kingdom.

Phylum refers to a type of coding error, while kingdom refers to a collection of phyla that share a common theme. Table 6 and Table 7 show the mapping of some of the existing hands-on labs to subsets of the kingdoms and phyla.

Kingdoms	Phylum	SWEET	Others
Input validation	Buffer Overflow		(Williams 2013)
	Command Injection		Security First:online banking (email injection)(Chu et al.2009)
	Cross-Site Scripting	Web-Application Threat Assessment Module	Tunestore (Chu et al.2009)Security First: online banking (Chu et al. 2009)
	Format String		
	Integer Overflow		
	SQL Injection	Web-Application Threat Assessment Module	Tunestore (Chu et al. 2009) Security First: online banking (Chu et al. 2009)
API Abuse	Directory Restriction Often Mis-used: Privilege Management		NCCure Health Care Systems:HR Benefit Management (Chu et al.2009)
Security Features	Missing Access Control		Tunestore (Chu et al.2009)Security First: online banking (XSRF) (Chu et al. 2009)
	Password Management		Tickle~Mail (Chu et. al2009) Tickle~Shop (Chu et. Al 2009)
Time and State	File Access Race Condition: TOCTOU		
Error Handling			Security First: online banking (XSRF) (Chu et al. 2009)

Table 7. The Mapping of Hands-on Labs to the Taxonomy of Coding Errors 2

4. Resources For Secure Software Engineering Curricla

During the past decade, industry and open source community have developed rich resources based on which secure software engineering curricula can be built. Some of the major resources are briefly described below.

4.1 Microsoft's Trustworthy Computing Initiatives

Microsoft started the Trustworthy Computing initiative in 2002 to improve public trust in its own commercial offerings. One of the outcomes of this initiative is the Microsoft Security Development Lifecycle (MS SDL) (Howard & Lipner 2005). MS SDL

series of security-focused activities and deliverables to each of the following phases of standard Microsoft's software development process: requirements, design, implementation, verification, release, and the support and servicing phase.

4.2 Build Security In

Build Security In (BSI) is a software assurance initiative sponsored by the Department of Homeland Security (US-CERT 2014). It includes rich resources such as best practices, knowledge and tools that can be used by software developers, architects, and security practitioners to build security into software in every phase of its development. McGraw (2006) introduces the process of applying a set of seven software security best practices called touchpoints into various software artifacts. The seven touch points are: code review, architectural risk analysis, penetration testing, risk-based security tests, abuse cases, security requirements, and security operations. They can be applied to such artifacts as code, design and specification, system in its environment, units and system, requirements and use cases, requirements, and fielded system. This touchpoint method can be applied to any software development process to integrate software security best practices into software development lifecycle.

4.3 CERT Software Assurance

As part of the Software Engineering Institute (SEI) at Carnegie Mellow University, the Computer Emergency Response Team (CERT) program's Secure Coding Initiative (SEI 2014b) has developed secure coding standards including "The CERT C Secure Coding Standard", "Secure Coding in C and C++", "The CERT Oracle Secure Coding Standard for Java", etc. The CERT Secure Coding Initiative also produced the Source Code Analysis Laboratory (SCALe) which offers conformity assessment of software to CERT secure coding standards, tools and libraries that help software developers reduce the number of vulnerabilities in their code.

4.4 OWASP

OWASP (2014b) is a world-wide non-profit organization that includes a collaborative community divided into local chapters. The community works to produce tools and documents to improve the security of software. OWASP has published good practice guides and tools on three areas: protection, detection and life-cycle security, such as the OWASP Top Ten, OWASP Testing Guide, OWASP Code Review Guide, Software Assurance Maturity Model, OWASP Anti-Samy Java Project, OWASP WebScarab Project, etc. The OWASP website also includes presentation slides and videos of presentations at OWASP conferences concerning application security.

OWASP OpenSAMM or SAMM project (Software Assurance Maturity Model) is a framework to help organizations to create software security program (OPENSAMM 2014). OpenSAMM defines security practices for four core business functions of software development: governance, construction, verification and deployment. An organization can fulfill a given security practice at three levels: 1) initial understanding and ad hoc provision of the practice; 2) increased efficiency and/or effectiveness of the practice; and 3) comprehensive mastery of the practice at scale. For each level, SAMM defines objective, activities, results, success metrics, costs, personnel, and related levels. An organization can improve its assurance program iteratively in phases by selecting security practices to improve, and achieving the next objectives in each practice by performing the corresponding activities at the specified success metrics. SAMM also includes assessment worksheets for each security practice and roadmap templates for typical kinds of organizations.

The OWASP CLASP project (OWASP 2014c; Graham 2006) provides a set of process components that can be integrated into software development lifecycles. The CLASP process includes five high level views: concept view, role-based view, activity assessment view, activity implementation view, and vulnerability view. These views are broken into 24 activities which are related to project roles such as project manager, security auditor, etc. CLASP also defines seven best practices: 1) Institute awareness programs, 2)Perform application assessments, 3) Capture security requirements, 4) Implement secure development practices, 5) Build vulnerability remediation procedures, 6) Define and monitor metrics, and 7) Publish operational security guidelines. The 24 CLASP activities are distributed across these best practices. CLASP also provides an extensive wealth of resources that aid in planning, implementing and performing CLASP activities.

4.5 SAFE Code

SAFECode (SAFECode 2014) is a global, non-profit organization that aims to increasing trust in technology products and services through identifying and promoting best practices for vendor software assurance. Its members include leading industries such as Microsoft Corp., Intel Corp., Adobe Systems Inc., etc. The SAFECode website publishes papers on software security such as "Software Security Guidance for Agile Practitioners", "Fundamental Practices for Secure Software Development", "Software Assurance: An Overview of Current Industry Best Practices", etc.

4.6 SAMATE

The SAMATE (Software Assurance Metrics and Tool Evaluation) project is sponsored by U.S. Department of Homeland Security (DHS) National Cyber Security Division and NIST (NIST 2014). The goal of this project is to improve software assurance by developing tool evaluation methods, measuring the effectiveness of tools and techniques, and identifying gaps in tools and methods. The SAMATE project has developed tool specifications, test plans, and test sets for the following types of tools: source code security analyzers, web application vulnerability scanners, and binary code scanners. The SAMATE also created the SAMATE Reference Dataset which consists of over 1800 community-contributed test cases which encompass a wide variety of flaws, languages, platforms, and compliers.

In summary, there are a number of resources developed by industry and open sources communities on software security. Table 8 lists a roadmap of resources which can be used in courses discussed in Section 3.2.

Secure coding/programming • CERT Software Assurance: The CERT C secure coding standard, Secure coding in C and C++, The CERT Oracle secure coding standard for Java • SAFECode: Software security guidance for agile practitioners, fundamental practices for secure software development, software assurance: an overview of current industry best practices. • OWASP code review guide • OWASP anti-Sammy Java project
Vulnerability Assessment/Software Security Testing • CERT Software Assurance: Source Code Analysis Laboratory (SCALe) • SAMATE: source code security analyzers, web application vulnerability scanners, and binary code scanners • OWASP testing guide, OWASP WebScarab project, OWASP WebGoat
Secure Software Engineering • Build Security In: best practices, knowledge, tools • Microsoft's Trustworthy Computing Initiatives: Microsoft Security Development Lifecycle (MS SDL) • Software Assurance Community Resources: security practice, methodologies, technologies • SAFECode: Fundamental practices for secure software development, Software Assurance-current industry best practices • OWASP: Open software assurance maturity model (SAMM) • OWASP CLASP

Table 8. Roadmap of Resources in Software Security

5. Conclusion

Since the majority of information security vulnerabilities are caused by software defects, it is important to provide college students as well as software professionals with education and training on software security. The Secure Software Assurance Common Body of Knowledge defined by DHS can be used to guide the development of education and training curriculum related to software assurance. The Software Assurance Reference Curriculum developed by CMU includes a core body of knowledge which can be the foundation of a Masters of software assurance standalone degree program or a track in software assurance. ACM/IEEE Draft Computer Science Curriculum includes secure software engineering knowledge units which guide the integration of software security education into a computer science undergraduate curriculum.

Secure software engineering curriculum can be developed in three approaches: (1) Concentrated courses; (2) Diffusion through curricula; (3) Concentration/degree program. Example programs, courses and teaching modules are available, and are helpful for educators to develop secure software engineering curriculum. Teaching modules in the areas of secure coding, software threat and threat analysis, secure application development, and software security and secure software engineering have been developed. They can be used to incorporate secure software engineering knowledge into existing computer science curricula.

A number of hands-on exercises developed by universities and open source community allow students to apply software

security knowledge to solve real life problems. These hands-on exercises can be mapped to the taxonomy of coding errors proposed by (McGraw 2006). The rich resources provided by industry and open source community can also be used to develop courses in secure coding/programming, vulnerability assessment/software security testing, and secure software engineering.

References

[1]. ACM and IEEE-Computer Society Joint Task Force on Computing Curricula (2013). Computer Science Curricula 2013 Ironman Draft (Version 1.0). Retrieved July 23, 2014 from http://ai.stanford.edu/users/sahami/CS2013/ironman-draft/cs2013-ironman-v1.0.pdf

[2]. ACM Two-Year College Education Committee (2009). Computing Curricula 2009: Guidelines for Associate-Degree Transfer Curriculum in Computer Science. Retrieved July, 23 2014 from http://www.capspace.org/committee/CommitteeFileUploads/2009ComputerScienceTransferGuidelines.pdf

[3.] BSIMM (2014). The Building Security In Maturity Model. Retrieved July 23, 2014 from http://bsimm.com

[4]. Chen, L., Tao, L., Li, X., Lin, C (2010). A tool for teaching web application security. In: Proc. of the 14th Colloquium for Information Systems Security Education (CISSE 2010), Baltimore, MD, 17-24.

[5]. Chu, B., Stranathan, W., Cody, J., Peterson, J., Wenner, A., Yu, H (2009). Teaching secure software development with vulnerability assessment. In: Proc, of the 13th Colloquium for Information Systems Security Education (CISSE 2009), Seattle, Washington, 146-150.

[6]. Cooper, S., Nickell, C., Pérez, L.C., Oldfield, B., Brynielsson, J., Gökce, A.G., Hawthorne, E.K., Klee, K.J., Lawrence, A., Wetzel, S (2010). Towards Information Assurance (IA) Curricular Guidelines. In: Proc. of the 2010 ITiCSE working group reports (ITiCSE-WGR '10), Clear, A., Dag, L.R (Eds.). ACM: New York, NY. 49-64.

[7]. European Union Agency for Network Information and Information Security (ENISA) (2011). Secure Software Engineering Initiatives: Listing SSE Initiatives Across Europe and Abroad. Retrieved July 23, 2014 from http://www.enisa.europa.eu/activities/Resilience-and-CIIP/critical-applications/secure-software-engineering/secure-software-engineering-initiatives

[8]. Graham, D (2006). Introduction to the CLASP Process. Retrieved July 23, 2014 from https://buildsecurityin.us-cert.gov/articles/best-practices/requirements-engineering/introduction-to-the-clasp-process

[9]. Howard M., Lipner, S (2005). The Trustworthy Computing Security Development Lifecycle. Retrieved October 25, 2012 from http://msdn.microsoft.com/en-us/library/ms995349.aspx

[10]. Howard, M., Lipner, S (2006). The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software. Redmond, Washington: Microsoft Press.

[11]. IEEE Computer Society (2010). Computer Society Recognizes Master of Software Assurance Curriculum. Retrieved July 23, 2014 from http://www.computer.org/portal/web/pressroom/20101213MSWA

[12]. McGraw, G (2006). Software Security: Building Security In. Crawfordsville, Illinois: Addison-Wesley Professional.

[13]. Mead, N.R., Allen, J.H, Ardis, M.A., Hilburn, T.B, Kornecki, A.J., Linger, R.C., McDonald, J (2010a). Software Assurance Curriculum Project Volume I: Master of Software Assurance Reference Curriculum. Technical Report CMU/SEI-2010-TR-005. Software Engineering Institute, Carnegie Mellon University. Retrieved July 23, 2014 from http://www.sei.cmu.edu/reports/10tr005.pdf

[14]. Mead, N.R., Allen, J.H., Ardis, M.A., Hilburn, T.B, Kornecki, A.J., Linger, R.C., McDonald, J (2011a). Software Assurance Curriculum Project Volume III: Master of Software Assurance Course Syllabi. Technical Report CMU/SEI-2011-TR-013. Software Engineering Institute, Carnegie Mellon University. Retrieved July 23, 2014 from http://www.dtic.mil/dtic/tr/fulltext/u2/a549397.pdf

[15]. Mead, N.R., Hawthorne, E.K, Ardis, M (2011b). Software Assurance Curriculum Project Volume IV: Community College Education, Technical Report CMU/SEI-2011-TR-017. Software Engineering Institute, Carnegie Mellon University. Retrieved July 23, 2014 from http://www.sei.cmu.edu/reports/11tr017.pdf

[16]. Mead, N.R., Hilburn, T.J, Linger, R.C (2010b). Software Assurance Curriculum Project Volume II: Undergraduate Course Outlines. Technical Report CMU/SEI-2010-TR-019. Software Engineering Institute, Carnegie Mellon University. Retrieved July 23, 2014 from http://www.sei.cmu.edu/reports/10tr019.pdf

[17]. National Institute of Standards and Technology (NIST) (2014). SAMATE- Software Assurance Metrics and Tool Evaluation.

Retrieved July 23, 2014 from http://samate.nist.gov

[18]. Northern Kentucky University (NKU) (2014). Graduate Certificate in Secure Software Engineering. Retrieved July 23, 2014 from http://informatics.nku.edu/departments/computer-science/programs/gcsse.html

[19]. OPENSAMM (2014). Software Assurance Maturity Model. Retrieved July 23, 2013 from http://www.opensamm.org

[20]. OWASP (2014a). Category: OWASP WebGoat Project. Retrieved July 23, 2014 from https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project

[21]. OWASP (2014b). Welcome to OWASP. Retrieved July 23, 2014 from https://www.owasp.org/index.php/Main_Page

[22]. OWASP (2014c). OWASP Clasp Project. Retrieved July 23, 2014 from https://www.owasp.org/index.php/Category:OWASP_CLASP_Project

[23]. Pace University (2014). Secure Web Development Teaching Modules, Retrieved July 23, 2014, from http://csis.pace.edu/~lchen/sweet/

[24]. Purdue University (2014). CS 390S: Secure Programming. Retrieved July 23, 2014 from http://www.cs.purdue.edu/homes/cs390s/

[25]. SAFECode (2014). SAFECode/security engineering training. Retrieved July 23, 2014 from http://www.safecode.org/index.php

[26]. Software Assurance (SwA) Workforce Education and Training Working Group (2007). Software Assurance: A Curriculum Guide to the Common Body of Knowledge to Produce, Acquire, and Sustain Secure Software. Retrieved July 23, 2014 from https://buildsecurityin.us-cert.gov/sites/default/files/publications/CurriculumGuideToTheCBK.pdf

[27]. Software Engineering Institute (SEI) at Carnegie Mellon University (2014a). Curricula: Software assurance Materials and Artifacts. Retrieved July 23, 2014 from http://www.cert.org/curricula/lecture-materials-and-artifacts.cfm

[28]. Software Engineering Institute (SEI) at Carnegie Mellon University (2014b). Secure Coding. Retrieved July 23, 2014 from https://buildsecurityin.us-cert.gov/

[29]. Stevens Institute of Technology (2013). New Offering: Software Assurance. Retrieved July 23, 2014 from http://www.stevens.edu/sse/academics/graduate/graduate-certificates/software-assurance

[30]. Syracuse University (2014). SEED: Developing Instructional Laboratories for Computer Security Education. Retrieved July 23, 2014 from http://www.cis.syr.edu/~wedu/seed/

[31]. ThreadStrong (2014). Application Security Training. Retrieved July 23, 2014 from http://www.threadstrong.com/index.html

[32]. Towson University (2014). Security Injections @Towson. Retrieved July 23, 2014 from http://cis1.towson.edu/~cssecinj/

[33]. US-CERT (2014). Build Security In: Setting a higher standard for software assurance. Retrieved July 23, 2014 from https://buildsecurityin.us-cert.gov/bsi/home.html

[34]. Walden, J., Frank, C.E (2006). Secure software engineering teaching modules. In: Proc, of the 3rd annual conference on Information security curriculum development (InfoSecCD '06). ACM: New York, NY, USA. 19-23. http://doi.acm.org/10.1145/1231047.1231052

[35]. Williams, K (2013). Teaching Stack Overflow. Retrieved September 8, 2013 from http://williams.comp.ncat.edu/overflow/Teaching.html

[36]. Yuan, X., Hernandez, J., Waddell, I., Chu, B., Yu, H (2012a). Hands-on Laboratory Exercises for Teaching Software Security. In: Proc. of the 16th Colloquium for Information Security Education (CISSE 2012), Lake Buena Vista, Florida. 15-20.

[37]. Yuan, X., Yu, H., Hernandez, J., Wadell, I (2012b). Integrating software security education into computer science curriculum. In: Proc. of the 11th IASTED International Conference on Software Engineering, Crete, Greece. 15-22.

CALL FOR PAPERS

SSH 2016: The 4th International Workshop on Service Science for e-Health co-located with 18th IEEE International Conference on e-Health Networking, Application & Services (HEALTHCOM 2016)
Munich, GERMANY – September 14-17, 2016
http://www.ssh2016.pwr.edu.pl/

Objectives and Topics

Recent technological solutions in computing and networking have largely made remote health services accessible worldwide

However, the key requirements for making a full potential of any healthcare systems include ubiquitous access to medical services, their personalization encompassing specific functionalities, broadly understood security of the patient's personaldata,openness to new networking technologies and techniques for the purpose of flexible management of the quality of service (QoS) and the considerable experience of the healthcare systems' developers.

Service science is an emerging interdisciplinary approach to design, implement and evaluate complex service systems. It integrates science with ICT technologies and business to provide a value added domain-specific applications of service-based systems.

The application of service science to the e-Health area is a natural tendency to build platforms, software which meet both the current and future demands of the healthcare domain.

The core objective of the workshop is to bring together delegates from academia, industry and healthcare business to present recent advances in the field of e-Health. High quality interdisciplinary papers that present innovative solutions in service science for healthcare systems are the most welcome. Prospective authors are invited to submit their original contributions covering completed or ongoing work related to the area of service science for healthcare.

The topics of interest include, but are not limited to:

- e-Health services design and implementation
- Business models for e-Health services' delivery
- Cloud computing for e-Health
- Internet of Things for e-Health
- Middleware for e-Health services
- Models and methods for decision making support in thehealthcare domain
- Network architectures for e-Health
- Service-based e-Health systems
- Virtual and augmented reality for e-Health
- Wireless access to e-Health services
- Mobile Health solutions
- ICT-Mediated medical information exchange

Important dates

Paper submission deadline: 30 June 2016

Notification of acceptance: 31 July 2016
Camera-ready papers: 31 August 2016
Conference date: 14-17 September2016

Submission & Proceedings

• Prospective authors are invited to submit their papers using the EDAS System at https://edas.info newPaper.php?c=22498&track=80455

• A full paper should not have more than five (5) IEEE style pages including the results, figures and references.

• All the papers will be reviewed using the standard reviewing procedure (by at least 2 independent anonymous reviewers).

• Accepted papers will be published on IEEE Xplore http:/ /ieeexplore.ieee.org/

• Authors of selected papers will be invited to submit extended versions of their papers to the following international journals:

• Artificial Intelligence in Medicine, Elsevier, IF (2014):2.019 http://www.journals.elsevier.com/artificialintelligence- in-medicine/

• Statistical Methods in Medical Research, SAGE Publishing, IF (2014): 4.472, http://smm.sagepub.com/

• Computers in Human Behaviour, Elsevier, IF (2014): 2.694 http://www.journals.elsevier.com/computers-in-humanbehavior

Eleventh International Conference on Digital Information Management (ICDIM 2016) Porto, Portugal September 19-21, 2016 (www.icdim.org)

Technically and financially co-sponsored by IEEE Technology Engineering Management Society

Proceedings will be indexed in IEEE Xplore

Following the successful earlier conferences at Bangalore (2006), Lyon (2007), London (2008), Michigan (2009) , Thunder Bay (2010), Melbourne (2011), Macau (2012), Islamabad (2013) Thailand (2014) and Jeju (2015) the eleventh event is being organized at Porto in Portugal in 2016. The International Conference on Digital Information Management is a multi-subdomain conference on digital information management, science and technology. The principal aim of this conference is to bring people in academia, research laboratories and industry together, and offer a collaborative platform to address the emerging issues and solutions in digital information science and technology. The ICDIM intends to bridge the gap between different areas of digital information management, science and technology. This forum will address a large number of themes and issues. The conference will feature original research and industrial papers on the theory, design and implementation of digital information systems, as well as demonstrations, tutorials, workshops and industrial presentations.

The 11th International Conference on Digital Information Management will be held during September 19-21, 2016 at Porto in Portugal.

The topics in ICDIM 2016 include but are not confined to the following areas.

- Information Retrieval
- Data Grids, Data and Information Quality
- Big Data Management
- Temporal and Spatial Databases
- Data Warehouses and Data Mining
- Web Mining including Web Intelligence and Web 3.0•E -Learning, eCommerce, e-Business and e-Government
- Natural Language Processing
- XML and other extensible languages
- Web Metrics and its applications
- Enterprise Computing
- •Semantic Web, Ontologies and Rules
- Human-Computer Interaction
- Artificial Intelligence and Decision Support Systems
- Knowledge Management
- Ubiquitous Systems
- Case Studies on Data Management,
- Monitoring and Analysis
- Security and Access Control
- Information Content Security
- Mobile, Ad Hoc and Sensor Network Security
- Distributed information systems
- Information visualization

• Web services

• Quality of Service Issues

• Multimedia and Interactive Multimedia

• Image Analysis and Image Processing

• Video Search and Video Mining

• Cloud Computing

ICDIM 2016 has the following co-located workshops Workshops Fifth Workshop on Advanced Techniques on Data Analytics and Data Visualization Fourth IEEE International Workshop on Data Management Second Workshop on Internet of Things Second Workshop on Big Data Mining Second Workshop on Cluster Computing Second Workshop on Intelligent Information Systems.

Proceedings

• All the accepted papers will appear in the proceedings published by IEEE.

• All papers will be fully indexed by IEEE Xplore.

• All the ICDIM papers are indexed by DBLP.

• Peer to Peer Data Management

• Interoperability

• Mobile Data Management

• Data Models for Production Systems and Services

• Data Exchange issues and Supply Chain

• Data Life Cycle in Products and ProcessesModified version of the selected papers will appear in the special issues of the following peer reviewed journals

1.Journal of Digital Information Management (SCOPUs/ EI)

2. Journal of Electrical Systems

3. Recent Advances in Electrical & Electronic Engineering

4. International Journal of Web Applications (IJWA)

5. International Journal of Information Technology and Web Engineering (IJITWE)

6. International Journal of Emerging Sciences (IJES)

7. International Journal of Grid and High Performance Computing (IJGHPC) (Scopus and EI Indexed)

8. International Journal of Computational Science and Engineering (Scopus and EI Indexed)

9. International Journal of Big Data Intelligence

10. International Journal of Applied Decision Sciences (Scopus/EI)

11. International Journal of Management and Decision Making (Scopus/EI)

12 International Journal of Strategic Decision Sciences

13. International Journal of Enterprise Information Systems (Scopus/EI

Important Dates

Full Paper Submission - July 10, 2016

Notification of Authors - August 1, 2016

Registration Due -	September 1, 2016
Camera Ready Due -	September 1, 2016
Workshops/Tutorials/Demos -	September 20, 2016
Main conference -	September 19-21, 2016

SUBMISSIONS AT http://www.icdim.org/submission.html

Committee

General Chair

Ramiro Sámano Robles, Instituto Superior de Engenharia do Porto Rua, Portugal

Program Chairs

Arun Pujari, Central University of Rajasthan, India Antonio J. Tallón-Ballesteros, University of Seville, Spain

Co-Chairs

Robert Bierwolf, IEEE TEMS, Netherlands Imran Bajwa, The Islamia University of Bahwalpur, Pakistan Feliz Lustenberger, Espros Photonics Corporation, Switzerland Francesco Piccialli, University of Naples "Federico II", Italy

Workshop Chair

Simon Fong, University of Macau, Macau

Email: conference at icdim.org

SUBMISSIONS AT http://www.icdim.org/submission.html

The Eighth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2017)
Universidad Aut Noma de Ciudad Juarez, Juarez,, Mexico
March 29-31, 2017
www.socio.org.uk/icadiwt

The Eighth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2017) is a forum for scientists, engineers, and practitioners to present their latest research results, ideas, developments and pplications in the areas of Computer Communications, Communication networks, Communication Software Communication Technologies and Applications, and other related themes.

This conference (ICADIWT Edition VIII) will include presentations of contributed papers and state-of-the-art lectures by invited keynote speakers.

Topics

This conference welcomes papers address on, but not limited to, the following research topics:
Internet Communication
Internet Technologies
Web Applications
Internet Software
Data Access and Transmission
Digital Communication Software
Digital Networks
Web Communication Interfaces
Internet Content Processing
Internet of Things
Internet of Everything
Data Communication
Databases and applications
Web Systems Engineering Design
Intelligent Agent Systems
Semantic Web Studies
Adaptive Web applications and personalization
Navigation and hypermedia

Important Dates

Submission of papers February 08, 2017
Notification February 15, 2017
Camera ready February 28, 2017
Registration February 28, 2017
Conference Dates March 29-31, 2017

Program Committees

Honorary Chairs (To be updated)

General Chairs

Ricardo Rodriguez Jorge, (Autonomous University of Ciudad Juarez, Mexico)
Osslan Osiris Vergara (Autonomous University of Ciudad Juarez, Mexico)
VianeyGpe. Cruz (Autonomous University of Ciudad Juarez, Mexico)