Infusing Principles and Practices for Secure Computing Throughout an Undergraduate Computer Science Curriculum

Jean R. S. Blair
United States Military Academy
West Point, New York, USA
Jean.Blair@westpoint.edu

Christa M. Chewar
United States Military Academy
West Point, New York, USA
Christa.Chewar@westpoint.edu

Rajendra K. Raj
Rochester Institute of Technology
Rochester, New York, USA
Rajendra.K.Raj@rit.edu

Edward Sobiesk
United States Military Academy
West Point, New York, USA
Edward.Sobiesk@westpoint.edu

ABSTRACT

In recent years, all computing disciplinary communities and curricular guidelines have increased their expectations of and requirements for incorporating cybersecurity into their discipline. For computer science, this has been a daunting task for a number of reasons, including the fast-paced evolution and expansion of the discipline, the perceived challenge of finding space in the curriculum, and the difficulty of selecting the best content.

This paper takes the position that infusing security concepts pervasively into an undergraduate Computer Science program is a crucial and attainable best practice. A five-step methodology is presented to incorporate cybersecurity into a traditional computer science curriculum in a way that maintains disciplinary integrity without adding significant new curricular content. This methodology is consistent with the philosophy and recommendations of the latest computer science and cybersecurity curricular guidelines. The paper also illustrates the application of these techniques to a typical Computer Science program.

CCS CONCEPTS

• **Social and professional topics** → **Computer science education; Model curricula; Accreditation.**

KEYWORDS

Computer science education; cybersecurity education; security education; curriculum development.

ACM Reference Format:
Jean R. S. Blair, Christa M. Chewar, Rajendra K. Raj, and Edward Sobiesk. 2020. Infusing Principles and Practices for Secure Computing Throughout an Undergraduate Computer Science Curriculum. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20), June 15–19, 2020, Trondheim, Norway.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3341525.3387426

1 INTRODUCTION

This paper presents a philosophy and approach for infusing cybersecurity content into an undergraduate Computer Science (CS) program. Our central premise is that good computing practices already within a modern CS curriculum provide substantial elements of cybersecurity content, and hence it is not necessary to add significant new content. The key insight is that many CS programs are simply missing a set of guiding cybersecurity principles that are purposefully and prominently linked throughout the curriculum.

We seek to dispel the mindset that CS fundamentals and cybersecurity are disparate. Furthermore, while most CS educators would agree that cybersecurity should be included in the CS curriculum—and might even argue that cybersecurity is a civic responsibility—Wolffe [24] states the uncomfortable truth that many CS programs do not require security coursework mainly because CS faculty members do not know what is needed or how to teach it. This paper provides an effective methodology to address this shortcoming.

We advocate that cybersecurity content should be threaded throughout a curriculum. However, finding ways to add security content along with the multitude of other evolving CS disciplinary topics and requirements can be daunting. Although some CS programs found early success with specialized security electives, such as Digital Forensics and Cybersecurity Engineering, sacrificing elective choices intended to provide breadth and depth seemed inappropriate. Additionally, our view is that it is not sufficiently effective to add security topics at the periphery of a curriculum such as in a networking course, a database course, or a professional seminar. Security must be fully integrated and should be present wherever the preponderance of the curriculum is taught.

Three salient questions are addressed in this paper:

(1) What security principles are most important for a given CS program?
(2) What existing content from required CS courses already contributes to coverage of security?
(3) How should security principles and associated practices be cohesively applied across the breadth and depth of a given CS program to be more memorable and, through sufficient repetition, more nuanced?

The next section sets the stage with an overview of the CS and cybersecurity curricular guidelines that are most relevant for a modern CS curriculum. Section 3 describes our proposed methodology to infuse security throughout a "typical" CS program and demonstrates application of our approach. Section 4 highlights additional related work and the paper ends with some concluding remarks.

2 RELEVANT CURRICULAR BACKGROUND

Two significant curricular guidelines are most germane to this paper. Computer Science Curricula 2013 (CS2013) [5] took a major step forward to ensure that information security and assurance (the then common term for cybersecurity) would receive the needed attention within undergraduate CS curricula. In fact, CS2013 provides much of the foundation for integrating security across a modern undergraduate CS curriculum. Four years later, the cybersecurity community published Cybersecurity Curricula 2017 (CSEC2017) [15], providing guidelines for cybersecurity undergraduate education. The CSEC2017 curricular model includes a disciplinary lens through which cybersecurity knowledge and skills are gained. When viewed through a CS lens, CSEC2017 provides a complementary perspective of cybersecurity in a CS curriculum. The remainder of this section highlights key aspects of CS2013 and CSEC2017.

2.1 CS2013 and Cybersecurity

The Computer Science Curricula 2013 report (CS2013) [5] was written by a Joint Task Force with members from both the Association for Computing Machinery (ACM) and the IEEE Computer Society. Although the focus of the document is on recommended curricular guidelines for undergraduate CS programs in the form of a Body of Knowledge, the publication also includes descriptions of the processes and principles used in its creation, a description of 11 characteristics expected of CS graduates, discussions of introductory courses and institutional challenges, advice on how to migrate to CS2013, 84 course exemplars, and five full curriculum exemplars. As stated in the document, CS2013 was written to provide guidance, not to be a standard for evaluating a program [5]. CS2013 organizes its curricular recommendations into three categories: a CS program should cover 100% of the Core Tier-1 topics, around 80-100% of the Core Tier-2 topics, and additional elective content to provide breadth and depth. A number of "lecture hours" is given for each curricular topic, but there is no assumption that the developmental experiences are provided through lecture format.

The recommendations in this Body of Knowledge are designed to be an "elegant interplay between theory, software, hardware, and application" presented through 18 Knowledge Areas (KAs). Each KA is further divided into Knowledge Units (KUs), and each KU has a list of curricular topics as well as suggested learning outcomes.

The CS2013 Information Assurance and Security (IAS) KA is:

> ... the set of controls and processes, both technical and policy, intended to protect and defend information and information systems by ensuring their confidentiality, integrity, and availability, and by providing for authentication and non-repudiation.

The KA also includes the need to question the state and validity of any inherited data, systems, or networks.

The IAS KA consists of only nine hours of Tier-1 and Tier-2 recommended content covering foundation concepts in Security, Principles of Secure Design, Defensive Programming, Threats and Attacks, Network Security, and Cryptography. Importantly, though, the IAS KA identifies an additional 63.5 hours across 13 of the other 17 KAs that are cybersecurity best-practices and principles. Of these, 32 hours are Tier 1 topics and the remaining 31.5 are Tier 2. The implication of this structure is that 7/8 of the cybersecurity content in the CS2013 Body of Knowledge is drawn from the other, more traditional parts of a Body of Knowledge. It is this aspect of the KA that makes it appropriate and straight-forward to continue teaching CS as we have, but to infuse cybersecurity by explicitly making the connections to cybersecurity throughout the CS curriculum.

SDF/Development Methods *[10 Core-Tier1 hours]*
This unit builds the foundation for core concepts in the Software Engineering knowledge area, most notably in the Software Processes, Software Design and Software Evolution knowledge units.
Topics:

- Program comprehension
- Program correctness
  - Types of errors (syntax, logic, run-time)
  - The concept of a specification
  - Defensive programming (e.g. secure coding, exception handling)
  - Code reviews
  - Testing fundamentals and test-case generation
  - The role and the use of contracts, including pre- and post-conditions
  - Unit testing
- Simple refactoring
- Modern programming environments
  - Code search
  - Programming using library components and their APIs
- Debugging strategies
- Documentation and program style

Figure 1: The CS2013 Development Methods Knowledge Unit – 8 of its 10 hours also serve as Information Assurance and Security content [5].

As an illustration of this concept, the IAS KA designates that eight of the ten Core Tier-1 hours of the Development Methods KU from the Software Development Fundamentals KA additionally serve as IAS content; see Figure 1. This example reinforces our major points that (1) most good security practices are simply good computing practices and (2) cybersecurity should be taught across the entire curriculum wherever situations requiring security are encountered, not just as an add-on course that covers security as a side topic to the curriculum.

2.2 CSEC2017 and Computer Science

The Cybersecurity Curricula 2017 report (CSEC2017) was an outcome of the Joint Task Force of several computing organizations – ACM, IEEE Computer Society, the Association for Information Systems Special Interest Group on Information Security and Privacy, and the International Federation for Information Processing Technical Committee on Information Security Education. CSEC2017 describes the cybersecurity discipline and outlines a Thought Model that includes knowledge areas (KAs), crosscutting concepts, and a disciplinary lens. As shown in Table 1, the KAs organize the cybersecurity content, with the first five (Data, Software, System,

Table 1: CSEC 2017 Knowledge Areas [15]

Knowledge Area	Description
Data Security	Protection of data at rest, during processing, and in transit
Software Security	Development and use of software that reliably preserves the security properties of the protected information and systems
Component Security	Security aspects of the design, procurement, testing, analysis, and maintenance of components integrated into larger systems
Connection Security	Security of the connections between components, both physical and logical
System Security	Security aspects of systems that use software and are composed of components and connections
Human Security	Study of human behavior in the context of data protection, privacy, and threat mitigation
Organizational Security	Protection of organizations from cybersecurity threats and managing risk to support successful accomplishment of the organizations' missions
Societal Security	Aspects of cybersecurity that broadly impact society as a whole

Component, and Connection) generally representing technical content, while the remaining three describe the Human, Organizational, and Societal dimensions. The KAs are not mutually exclusive; the crosscutting concepts, described in Table 2, provide the basis for interrelating knowledge areas into a "coherent view of cybersecurity" [15].

CSEC2017 describes a disciplinary lens as:

> ... the underlying computing discipline from which the cybersecurity program can be developed. The disciplinary lens drives the approach, depth of content, and learning outcomes resulting from the interplay among the topics. ... The application of the crosscutting concept and/or the level of depth taught within each knowledge unit may differ depending upon the disciplinary lens [15].

Using the CS disciplinary lens, CSEC2017 provides an additional perspective on how to integrate cybersecurity content into a CS curriculum. The eight KAs and six crosscutting concepts are viewed and applied from a CS perspective, providing technical and non-technical cybersecurity content appropriate for a CS curriculum. For example, consider the impact of the CS disciplinary lens on the Data Security KA. Here, the curriculum must include: encryption algorithms for securing data at rest; secure networking protocols for when data is in transit; access control techniques to ensure confidentiality and integrity; and fault tolerance to ensure availability.

3 INFUSING SECURITY INTO A TYPICAL COMPUTER SCIENCE CURRICULUM

To address the challenges described in Section 1 while maintaining consistency with the precepts laid out in CS2013 and insights from

Table 2: CSEC 2017 Crosscutting Concepts [15]

Concept	Description
Confidentiality	Limiting access to system data and information to authorized persons
Integrity	Assurance that the data and information are accurate and trustworthy
Availability	Data, information and systems are accessible
Risk	Potential for gain or loss
Adversarial Thinking	A thinking process that considers the potential actions of the opposing force working against the desired result
Systems Thinking	A thinking process that considers the interplay between social and technical constraints to enable assured operations

CSEC2017, this section presents the following five-step methodology for infusing cybersecurity into a typical CS curriculum:

(1) Identify the appropriate security principles for the program's constituents, taking into consideration the pervasive vulnerabilities and threats in the world today as well as existing recommendations and guidelines.
(2) Map existing curriculum content to the principles – identify where the curriculum already covers topics and practices related to and supporting your selected security principles.
(3) Identify the gaps.
(4) Determine to what extent and how the gaps will be filled.
(5) Purposefully connect and articulate the security principles and content throughout the curriculum.

The follow-on subsections will expand on and illustrate the steps in this methodology. In describing a generic traditional CS curriculum, we will use four threads rather than specifying courses in the generic curriculum. The threads group similarly focused required coursework in a typical CS curriculum. They are:

- Programming: programming fundamentals, data structures, and software development.
- Theoretical foundations: discrete math, CS theory, algorithms and complexity, and programming language concepts.
- Systems: computer organization and architecture, operating systems, networking and communication, and parallel and distributed computing.
- Computing in practice: information management, professional seminar, and a major project.

As an exemplar of this generic modern CS curriculum, the West Point CS program begins in the sophomore year and includes a typical CS 1 course that is taken during the first semester at the same time as cybersecurity fundamentals; the latter provides early exposure to the computing in practice and information management topics, as well as risk and adversarial thinking. The second semester coursework includes data structures, discrete math, and embedded systems with digital logic, while the third semester covers computer organization, programming language concepts, and networking. Building on this foundation is coursework in algorithms, CS theory, and software development and testing during the fourth semester. Seniors take operating systems and professional considerations, along with a few electives, and complete a year-long interdisciplinary capstone project.

3.1 Step 1: Identifying Security Principles

Each CS program will have its own constituents and associated areas of emphasis and/or distinctive topics. As such, the program should explore, select, and organize the security principles that provide the best fit. This subsection describes our selection for the typical, traditional CS program at West Point.

We considered security content from several sources including (a) CS2013 [5] and CSEC2017 [15], (b) the *Security Fundamental Principles* knowledge unit of the National Security Agency's Center for Academic Excellence in Cyber Operations criteria [8], (c) the Open Web Application Security Project's *Security by Design Principles* [14], and (d) a modernized form of Saltzer and Schroeder's security principles [18]. From these sources, we distilled and selected the following seven security principles.

(1) Confidentiality, Integrity, and Availability – the traditional security pillars
(2) Open Design – no security through obscurity
(3) Economy of Mechanism – simplest possible solution for smaller attack surface
(4) Least Privilege and Complete Mediation – user access to data/tools is as restricted as possible; every access to every resource must be validated
(5) Fail Safe Defaults – resource access is closed unless granted
(6) Defense in Depth – layered approach to resource access
(7) Psychological Acceptability – security cannot be overly difficult for users

To determine these principles, we selected those that worked best for our program and purposefully chose not to include other excellent candidates that are less important for our constituents. The CS community may benefit from further efforts to develop consensus on the most appropriate set of principles for textbooks and common use. Of course, each program would still be at liberty to include principles that would be most meaningful in their program's context and exclude those that are less appropriate. For instance, a program catering to the corporate or financial software industries might opt to include security principles related to mitigation of conflicts of interest under the Chinese Wall security model [9].

3.2 Steps 2-5: Integrating the Principles

Once the set of principles is established, the important process of mapping the principles to the CS curriculum must take place. Table 3 shows a step 2 mapping of relevant CS practices and concepts from each of the four curricular threads in the generic traditional CS program to our security principles. This particular mapping assumes previous introductory coverage of the following topics: malware, denial of service, social engineering, risks, threats, vulnerabilities, trust, social media, maintaining security updates, appropriate user interfaces, and trustworthiness of information.

A finer-grained mapping of a specific CS program might use courses rather than threads. It is interesting to note that for the mapped program, every required course would have at least a few practices/topics that it contributes to the overall security knowledge, skills, and attributes.

Addressing steps 3-5 for the program, one might determine that most of the needed security content was already present throughout the courses in the threads, although in response to the mapping, courses should be adjusted to explicitly frame and motivate the CS content, relating it to the security principles.

3.3 Example: Evolution of a Software Development Course at West Point

Steps 3-5 of the process will vary greatly and step 5 in particular may take time to reach a steady state as courses evolve and respond to changes in prerequisite and follow-on courses. To illustrate the evolution of a course in the West Point CS program over the past several years (both to update content and to integrate coverage of security topics), we look at what is now the Software Testing and Development course. It has long been an advanced programming topics course, with a general goal of preparing students with the knowledge and skills needed to be an effective team member in their capstone projects. Several years ago, the course was named Object-Oriented Concepts and included heavy coverage of design patterns to exercise ideas like inheritance, composition, encapsulation, abstraction, and concurrency, with the challenge of working on a medium-sized codebase that is iteratively developed over several weeks. Specification writing, front-end design, and documentation continue as supplemental themes, despite several language changes and inclusion of web and database integration.

After the publication of CS2013, an overall curriculum audit of our CS program suggested refocusing this course toward more general software development and testing topics. We deemed that the Programming Languages course's coverage of object-oriented concepts as a contrast to other programming styles was sufficient. This change allowed inclusion of new topics such as version control concepts and practice, test-driven development (TDD) and automated test harnesses, and an exposure to full-featured IDE tooling for debugging. To enhance web development topics, we took TDD a step further to continuous integration tools. While most of the design patterns were put aside, the theme of assuring software maintainability, especially to facilitate teamwork and agile development, remained a critical focus.

A few years later, when security principles and practices became required for all computing programs accredited by ABET [1], we took another look at fully integrating security topics into the Software Development and Testing course. Although it was easy to identify in this course at least ten Core-tier 1 hours from the CS2013 SDF/Development Methods knowledge unit (see Figure 1), we found it more difficult to convince external stakeholders and students there was enough security focus; the course topics simply were not in the security-focused language that they expected. There were a few quick wins, such as including code reviews that focused on OWASP checklists [14] and mention of language-specific vulnerabilities, but these felt tangential.

Our breakthrough came when we considered repackaging this content to place primary focus on our security principles and discussing the principles in the order that best supported the progression of topics in the course. In other words, a lesson that had

Table 3: Security Practices in CS Curricular Threads

Security Principle	Programming	Theoretical Foundations	Systems	Computing in Practice
Confidentiality Integrity Availability	Introduction to Concepts Program Correctness Maintainable Software		Redirecting Traffic Traffic Analysis Man-in-the-Middle Spoofing Sniffing	Confidentiality / Privacy Data Sanitization Accessibility for All Users Risks, Threats, and Vulnerabilities
Open Design	Program Comments / Style Modularity for Readability Variable Names Abstract Data Types Generics Software Reuse UML / Documentation Invariants Test-driven Development Remote Repositories Design Patterns	Algorithm Design for Simplicity Search Algorithms Programming Style Design Paradigms Design Patterns	System Architecture	Ethics of Vulnerability Disclosure Team Code Review
Economy of Mechanism	Coding Standards Design by Contract Single-purpose Classes Refactoring Composition vs. Inheritance Strategy Pattern	Coding Standards		
Least Privilege and Complete Mediation	Input Validation Garbage Collection Interpret vs. Compile Side Effects Limiting Inheritance Encapsulation Sequence Diagrams Abstraction	Non-Determinism from Parallelism Probability Interpret vs. Compilation Grammars and Type Checking Parsing, Syntax, Ambiguity Manual vs. Automated Memory Management Implementation of Loops and Recursion	Run-time Memory, the Call Stack and Heap Memory Management Memory Leaks DNS User Access	
Fail Safe Defaults	Use of APIs Exception Handling Unit Testing Types of Errors Constructor Options Option Types Test Case Generation Test-driven Development	Program Comprehension Type-safe Languages Integer Errors Code Reviews	Buffer Overflow Debugging Race Conditions Process Isolation Virtualization	Input Validation SQL Injection Defense Cross-site Scripting Vulnerabilities
Defense in Depth	Encryption Separation of Duties (MVC) Information Hiding	Permutations, Combinations True Randomness Choice of Language	Error Correcting Codes Fault Tolerance Correction Techniques Exceptions Intrusion Detection Honeypots Firewalls Boundary Security Use of Cryptography	SQL Injection Defense
Psychological Acceptability				Accessibility Social Models Empirical Testing

been titled "Design by Contract," which emphasizes single-purpose class design and minimization of side effects, supports the security principle of "Economy of Mechanism" so we re-titled the lesson and used the principle to motivate introduction of similar content.

After we began, this process became easier, but only after we carefully reflected upon the broad meaning of each security principle and identified its supporting course content. Some principles seemed like a stretch, but more than half were natural fits within each identified course. For example, we easily identified many course topics that supported Open Design. We became more confident after seeing the complete mapping of topics within the entire CS program to the security principles. Not only were we able to see a progression of topics within a given thread (i.e., all Programming thread courses had topics supporting Open Design), but we also found other courses that addressed principles that did not fit within the Software Testing and Development course. We also recognized the potential for repetition and nuance that came from grounding applicable course topics in introduction/reintroduction of the overarching principle. For instance, by the time students reached this course, they would have been exposed to the importance of Open Design at least three times, and they would hear about it again three more times the following year in Systems and Computing in Practice courses.

Our evolution occurred across several years for this course, roughly at the same pace that cybersecurity guidelines were initially published and popularized. Had we begun with the insights outlined in Section 3, the infusion of security principles might have taken a matter of months. Although, similar to the agile software development methodology, we advocate multiple iterations of our five step process.

3.4 Security in the RIT CS Curriculum

To meet the recent ABET requirement for coverage of security principles and practices [1], the RIT CS curriculum is being revised. Unlike West Point, RIT's program focuses on preparing graduates for a wide variety of careers, primarily in industry, and requires a cooperative education component. Albeit different in terms of courses, most topics required by RIT are also required at West Point, and are likely present in any modern CS curriculum.

In short, the proposed five-step methodology, along with the breakdown shown in Table 3, can directly be applied to RIT's CS program. Rather than covering artificial intelligence, data management, and parallel and distributed computing in other courses as at West Point, RIT has a full course for each topic. These courses add further opportunities to identify security topics in RIT's curriculum.

4 RELATED WORK

All six computing disciplines recognized by the ACM now have some form of cybersecurity content significantly integrated into their recommended curricular guidelines [4–7, 15, 17].

Several special editions of journals [10, 11, 20] have been devoted to cybersecurity education. Conferences and workshops focused on cybersecurity education have come into being, for example, the Colloquium for Information Systems Security Education (CISSE) [12]. In recent years, the annual ACM Technical Symposium on Computer Science Education (SIGCSE) [2] and the ACM Conference on IT Education (SIGITE) [3] have also had several papers, panels and workshops focused on cybersecurity. The focus however, with a few exceptions discussed below, has not been to extract and emphasize cybersecurity within existing CS curricula.

Various authors have explored how security content can be integrated into CS curricula, developing advice, best practices, and recommendations. Specifically, Null's seminal effort [16] looked at the entirety of Computing Curricula 2001–Computer Science [13] and suggested how security could be incorporated into CS, as it existed then. Much later, Siraj et al. [19] discussed an approach for integrating security across CS by exposing students to computer security concepts in several existing courses in their regular program of study. Taylor and Kaza showed how secure coding concepts could be injected into introductory CS courses [21], exposing CS students to integer overflow, buffer overflow, and input validation. Weiss et al. explored how CS educators could work as a community to teach cybersecurity [22] and subsequently integrate hands-on cybersecurity exercises into the CS curriculum [23].

5 FINAL REMARKS

This paper presented a philosophy and approach for infusing cybersecurity content into a traditional undergraduate CS program using a five-step methodology. The key aspects of our methodology are that programs should identify the cybersecurity principles they want to adopt, map their current curriculum against them, identify and mitigate gaps, and purposefully connect and articulate the security principles throughout their curriculum.

A central premise was that good computing practices already existing in the curriculum provide substantial cybersecurity content, which implies that it may not be necessary to add a significant amount of new content. We strongly advocate that good security practices are now simply good computing practices and that cybersecurity should be taught across an entire curriculum.

As for future work, we see cybersecurity becoming ubiquitous throughout CS education. Instead of asking where should I teach cybersecurity, the question may become: *where do I not teach cybersecurity – with the answer to this latter question presumably being almost nowhere*. Additionally, the weakest link in any attack surface will continue to be humans, and this will impact best practices and curricular content as well. Although cybersecurity already encompasses many tools, human-machine teaming (using AI) will grow ever more prevalent as will securing the many emerging technologies of the 21st Century, all of which will impact CS curricula. Finally, a much needed area of research is to explore educational pedagogy and content on the topic of the trade-offs between security and other goals.

ACKNOWLEDGMENTS

Blair, Raj and Sobiesk acknowledge innumerable cybersecurity education discussions with CSEC2017 co-authors Joseph J. Ekstrom, David Gibson and Allen Parrish. Raj acknowledges support by the National Science Foundation under Awards 1433736 and 1922169.

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the U.S. Military Academy, the Department of the Army, the Department of Defense, or the U.S. Government.

REFERENCES

[1] ABET, Inc. 2019. Criteria for Accrediting Computing Programs, 2019–2020. http://www.abet.org/wp-content/uploads/2018/11/C001-19-20-CAC-Criteria-11-24-18.pdf, Accessed: January 14, 2020.

[2] ACM SIGCSE 1970–. *SIGCSE Annual Technical Symposium*. ACM SIGCSE. https://sigcse.org/sigcse/events/symposia/index.html, Accessed January 19, 2020.

[3] ACM SIGITE 2000–. *SIGCSE Annual Conference*. ACM SIGITE. https://www.sigite.org/?cat=6, Accessed April 14, 2020.

[4] ACM/AIS Task Group on Information Systems Curricula. 2010. *Information Systems 2010*. Technical Report. ACM Press. https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2005-march06final.pdf, Accessed November 06, 2018.

[5] ACM/IEEE-CS Joint Task Force on Computing Curricula. 2013. *Computer Science Curricula 2013*. Technical Report. ACM Press and IEEE Computer Society Press. https://doi.org/10.1145/2534860 Accessed: November 14, 2017.

[6] ACM/IEEE-CS Task Group on Computer Engineering Curricula. 2016. *Computer Engineering Curricula 2016*. Technical Report. ACM Press and IEEE Computer Society Press. https://doi.org/10.1145/3025098 https://dx.doi.org/10.1145/30325098, Accessed November 06, 2018.

[7] ACM/IEEE-CS Task Group on Information Technology Curricula. 2017. *Information Technology Curricula 2017*. Technical Report. ACM Press and IEEE Computer Society Press. https://doi.org/10.1145/3173161 https://dl.acm.org/citation.cfm?id=3173161, Accessed November 06, 2018.

[8] US National Security Agency. 2019. Academic Requirements for Designation as a CAE in Cyber Operations. https://www.nsa.gov/Resources/Students-Educators/centers-academic-excellence/cae-co-fundamental/requirements/#m8, Accessed Jan 18, 2020.

[9] D. F. C. Brewer and M. J. Nash. 1989. The Chinese Wall security policy. In *Proceedings. 1989 IEEE Symposium on Security and Privacy*. IEEE, Oakland, CA, 206–214.

[10] Diana L. Burley. 2014. Cybersecurity Education, Part 1. *ACM Inroads* 5, 1 (March 2014), 41. https://doi.org/10.1145/2568195.2568210

[11] Diana L. Burley. 2015. Cybersecurity Education, Part 2. *ACM Inroads* 6, 2 (May 2015), 58. https://doi.org/10.1145/2746407

[12] CISSE 1997–. *The Colloquium for Information Systems Security Education (CISSE)*. CISSE. http://www.cisse.info/, Accessed January 19, 2020.

[13] Joint Task Force for Computing Curricula 2001. 2001. *Computing Curricula 2001: Computer Science–Final Report*. Technical Report. ACM Press and IEEE Computer Society Press. https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2001.pdf, Accessed January 19, 2020.

[14] OWASP Foundation. 2017. The Open Web Application Security Project (OWASP) Security by Design Principles. https://wiki.owasp.org/index.php/Security_by_Design_Principles Accessed Jan 18, 2020.

[15] Joint Task Force on Cybersecurity Education. 2017. *Cybersecurity Curricula 2017*. Technical Report. ACM, IEEE-CS, AIS SIGSEC, and IFIP WG 11.8. https://doi.org/10.1145/3184594

[16] Linda Null. 2004. Integrating Security across the Computer Science Curriculum. *J. Comput. Sci. Coll.* 19, 5 (May 2004), 170–178.

[17] ACM/IEEE-CS Task Group on Software Engineering Curricula. 2014. *Software Engineering 2014*. Technical Report. ACM Press and IEEE Computer Society Press.

[18] Jerome H. Saltzer and Michael .D. Schroeder. 1975. The Protection of Information in Computer Systems. In *Proceedings of the IEEE*, Vol. 63. IEEE, New York, 1278–1308.

[19] Ambareen Siraj, Blair Taylor, Siddarth Kaza, and Sheikh Ghafoor. 2015. Integrating Security in the Computer Science Curriculum. *ACM Inroads* 6, 2 (May 2015), 77–81. https://doi.org/10.1145/2766457

[20] A. Sobel, A. Parrish, and R. K. Raj. 2019. Curricular Foundations for Cybersecurity. *Computer* 52, 3 (March 2019), 14–17. https://doi.org/10.1109/MC.2019.2898240

[21] Blair Taylor and Siddharth Kaza. 2016. Security Injections@Towson: Integrating Secure Coding into Introductory Computer Science Courses. *ACM Trans. Comput. Educ.* 16, 4, Article 16 (June 2016), 20 pages. https://doi.org/10.1145/2897441

[22] Richard Weiss, Ambareen Siraj, Jens Mache, Elizabeth Hawthorne, Blair Taylor, Siddharth Kaza, and Michael E. Locasto. 2017. Building and Supporting a Community of CS Educators Teaching Cybersecurity in 2017 (Abstract Only). In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 732. https://doi.org/10.1145/3017680.3022370

[23] Richard Weiss, Ambareen Siraj, Jens Mache, Blair Taylor, Siddharth Kaza, Ankur Chattopadhyay, and Michael E. Locasto. 2018. Integrating Hands-on Cybersecurity Exercises into the Curriculum in 2018: (Abstract Only). In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 1070. https://doi.org/10.1145/3159450.3162195

[24] Josephine Wolffe. 2016. Why Computer Science Programs Don't Require Cybersecurity Classes. https://slate.com/technology/2016/04/why-computer-science-programs-dont-require-cybersecurity-classes.html, Accessed January 15, 2020.