Software Assurance Curriculum for Two-Year Associates in Applied Science Degree

Carol Woody, Ph.D.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

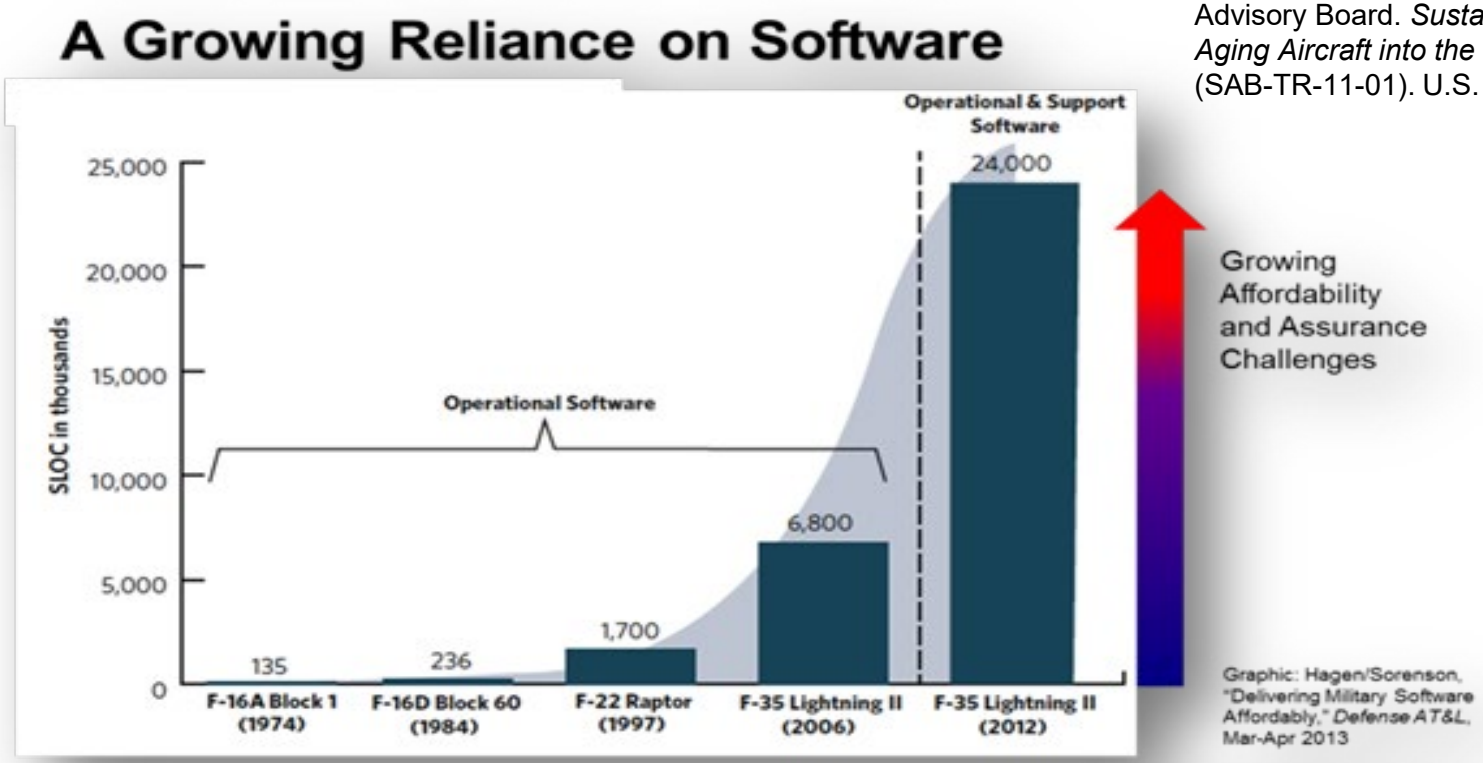
This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM18-1161

Importance of Software Assurance to Today's Environment

Software Reliance is Rapidly Expanding

Source: U.S. Air Force Scientific Advisory Board. *Sustaining Air Force Aging Aircraft into the 21st Century* (SAB-TR-11-01). U.S. Air Force, 2011.



Software as % of total system cost

1997: 45% → 2010: 66% → 2024: 88%

From 1997 to 2012, software industry production grew from $149 billion to $425 billion

Anyone Can Write Software but is it Good?

How To Raise The Next Zuckerberg: 6 Coding Apps For Kids

http://readwrite.com/2013/04/19/how-to-raise-the-next-zuck-6-coding-apps-for-kids/

TYNKER - We Empower KIDS to Become Makers

https://www.tynker.com/

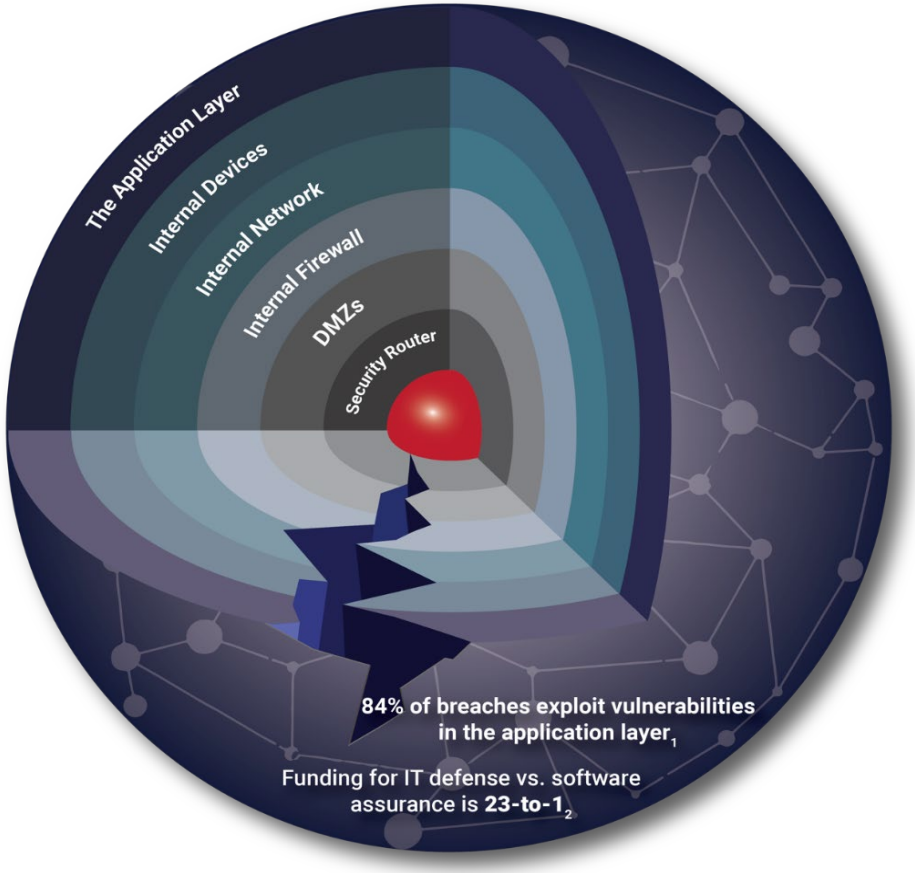
How and Why to Teach Your Kids to Code

http://lifehacker.com/how-and-why-to-teach-your-kids-to-code-510588878

Best-in-class code: <600 defects per MLOC
Very good code: 600 to 1,000 defects per MLOC
Average quality code: 6000 defects per MLOC

Up to 5% of defects are vulnerabilities

84% of Security Breaches Exploit the Software Applications



Breaking this cycle will require engineering of the software we use to handle the realities of the operational environment. All fielded software needs good cybersecurity. However,

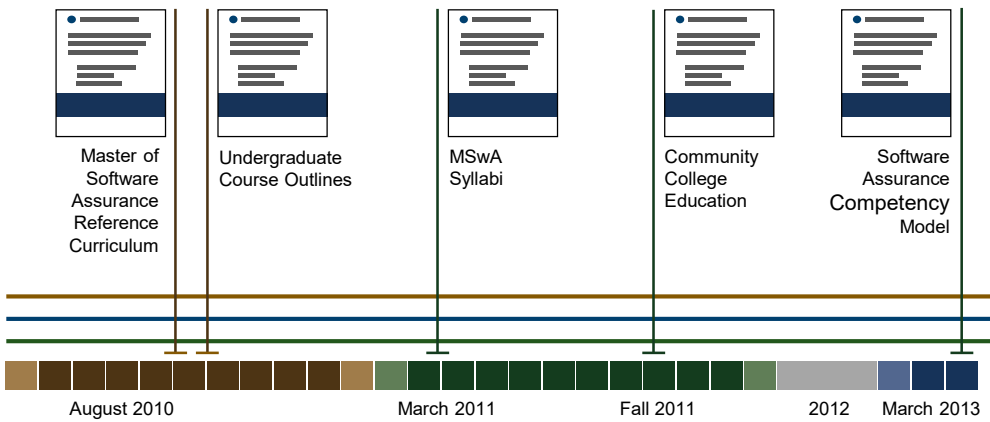
- "76% of U.S. developers use no secure application program process"[3]
- "More than 40% of software developers globally say that security isn't a top priority for them"[4]

1. Clark, Tim, *Most cyber Attacks Occur from this Common Vulnerability,* Forbes. 03-10-2015
2. Feiman, Joseph, *Maverick Research: Stop Protecting Your Apps; It's Time for Apps to Protect Themselves,* Gartner. 09-25-2014. G00269825
3. *Horvath, Mark, Neil MacDonald, Ayal Tirosh: Integrating Security Into the DevSecOps Toolchain, Gartner. 11-16-2017. G00334264*
4. Microsoft[1]– http://visualstudiomagazine.com/articles/2013/07/16/majority-of-us-devs-dont-practice-secure-coding.aspx

Preparing the Incoming Workforce to Handle Software Assurance

Software Assurance Curriculum Project

Goals: Develop software assurance curricula
Define transition strategies for implementation



Professional Society Recognition

IEEE acm

Community Outreach
- 20+ Published Papers
- 7 SEI reports
- 15+ talks, webinars, podcasts, media
- Thousands of downloads
- LinkedIn group of 500+ members
- Course materials and videos

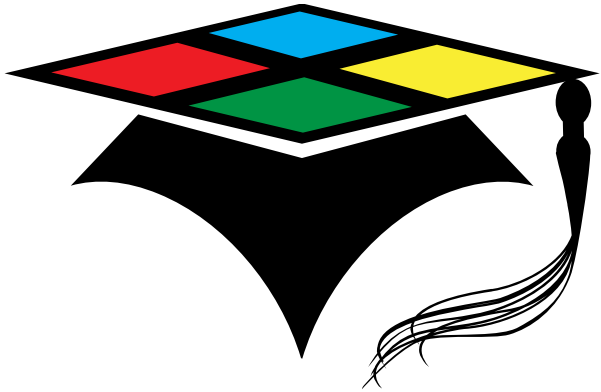
Integrated into course offerings
- Carnegie Mellon University
- Stevens Institute of Technology
- US Air Force Academy
- University of Detroit Mercy
- University of Houston
- $(ISC)^2$

Transition
- Degree offerings
  - MSwA Curriculum Design: Polytechnic University of Madrid
  - Community College Programs: Illinois Central College, Alamo Colleges, Lincoln Land Community College
- SwA Courses
  - Assurance Management
  - Assured Software Dev't 1
  - Exec Course

https://www.sei.cmu.edu/education-outreach/curricula/software-assurance/index.cfm

SwA Curriculum Project Objectives

Improve the state of software assurance education



Develop a Master of Software Assurance Reference Curriculum (Volume I)

Identify educational offerings at other levels

- Undergraduate (Volume II)
- MSwA Syllabi (Volume III)
- Community College (Volume IV)
- Integration with IS Curricula (Technical Note)

Professional Society Recognition

IEEE Recognition

The MSwA curriculum was recognized by the IEEE Computer Society. Its notification follows:

> *At the meeting of the IEEE Computer Society Board of Governors it was passed:*
>
> *MOVED, that the IEEE Computer Society Board of Governors recognizes the SEI CMU/SEI-2010-TR-005 Reference Curriculum as appropriate for a Masters Program in Software Assurance for a period of 5 years beginning in 1 August 2010.*
>
> ***Statement:*** The curriculum recommendation could contain a statement similar to *"The IEEE Computer Society recognizes this curriculum recommendation as appropriate for a Masters Program in Software Assurance,"* signifying that the Society considers it suitable for its stated purpose. If the curriculum recommendation is appropriate as a model for similar efforts, the statement should indicate that designation.

IEEE published an article about its recognition of the MSwA curriculum at http://www.computer.org/portal/web/pressroom/20101213MSWA.

ACM Recognition

The MSwA curriculum was also recognized by the Association for Computing Machinery (ACM) Education Board. This is identical to the IEEE recognition.

Community College Report

An ACM committee on two-year degree programs, led by Elizabeth Hawthorne, partnered with the SEI team to develop this report.

The report includes

- discussion of existing curricula related to software security that are suitable for community colleges
- target audience
- course outlines
- identification of resources

Community College Courses

Target audience: Students planning to transfer to a four-year program, students with prior undergraduate technical degrees who wish to become more specialized in software assurance

Courses:

- Computer Science I, II, and III
- Introduction to Computer Security
- Secure Coding
- Introduction to Assured Software Engineering

Students will not necessarily take all six courses, nor will the courses necessarily match courses in four-year colleges with similar titles.

Computer Science I

Topic	Bloom's Level
Secure coding (2 hours): data protection techniques of data encapsulation, information hiding and integrity, and strict data typing	A
Fundamental programming constructs (11 hours): basic syntax and semantics of a higher-level language; variables (scope and lifetime), types, expressions, and assignment; self-documentation; standard and file input/output; conditional and iterative control structures; structured decomposition; pseudo-random number generator	A
Fundamental algorithms and problem-solving (6 hours): problem-solving strategies; the role of algorithms in the problem-solving process; implementation strategies for algorithms; debugging strategies; the concept and properties of algorithms	A
Fundamental data structures (6 hours): primitive types, arrays, records, strings, references	A
Object-oriented principles (6 hours): abstraction, objects, classes, methods, parameter passing, encapsulation, inheritance, polymorphism	A
Program development (3 hours): program development phases, with emphasis on design, implementation, and testing and debugging strategies	A
Software tools and integrated development environment (IDE) (2 hours): compiling, interpreting, linking, executing, testing, and debugging	A
Programming languages (1 hour): comparison of object-oriented, procedural, functional programming	C
Human-computer interaction (1 hour): sound design concepts and fundamental graphical interface design; standard API graphics	C
Machine-level representation of data (1 hour): overview of the storage of instructions, numbers, and characters in a Von Neumann machine	C
Ethical conduct (1 hour): codes of ethics and responsible conduct; intellectual property, copyright, and plagiarism; "Ten Commandments for Computer Ethics"	C
Overview of operating systems (1 hour): role and purpose of operating systems; simple file management	C
Historical context of computing (1 hour): history of computing ideas, computing, and programming	K

Computer Science II

Topic	Bloom's Level
Secure coding (3 hours): buffer overflows; memory leaks; malicious code; unauthorized and back-door access; security-aware exception handling	A
Software development (4 hours): software life cycle; test case design; software tools; debuggers and simulators; characteristics of maintainable software; program code verification and data validation; software inspection	A
Object-oriented programming (7 hours): encapsulation and information hiding; inheritance; class hierarchies; polymorphism; abstract and interface classes	A
Object-oriented design and modeling (5 hours): class constructors and destructors; abstract data types (ADTs); reusable software components; APIs; modeling tools; class diagrams	A
Intermediate programming constructs (3 hours): cohesion and decoupling; assertions, including pre/post conditions and loop invariants; software reuse; self-documentation	A
Intermediate computing algorithms (5 hours): searching; sorting; recursive algorithms; complexity of algorithms	A
Intermediate data structures (7 hours): built-in; programmer-created; dynamic	A
Event-driven programming (4 hours): graphics API; event creation; event-handling methods; exception handling	A
Human-computer interaction (2 hours): sound design concepts; interfaces between people and technology	C
Simple database integration (1 hour): database I/O; embedded SQL queries; SQL injection	C
Societal and professional issues (1 hour): computing and the internet; social impact of computing; privacy	C

Computer Science III

Topic	Bloom's Level
Software assurance (3 hours): conformance with assurance coding standards and practices, trustworthiness, and predictable execution testing; quality reviews; engineering and security tradeoffs; risks and liabilities of computer-based systems; fault prevention in software life-cycle stages; intentional and unintentional software security vulnerabilities.	A
Formal computing algorithms (8 hours): efficiency of various sorting and searching algorithms; hashing; collision-avoidance strategies; binary search trees; depth- and breadth-first traversals; shortest-path algorithms; minimum spanning tree; transitive closure; topological sort	A
Canonical data structures (7 hours): stacks; queues; linked lists; hash tables; trees; graphs	A
Recursion (7 hours): recursive mathematical functions; divide-and-conquer, first-and-rest, and last-and-rest strategies; backtracking; recursion with linked lists; trees; graphs	A
Software reuse (3 hours): design patterns; parametric polymorphism (templates or generics); code libraries; container classes and iterators	A
Human-computer interaction (2 hours): universal principles; human-centered considerations; usability testing and verification; design tradeoffs; secure user interfaces	C
Software engineering (4 hours): standard approaches and implementation tools for analysis and design; measurement and metrics; software life-cycle stages, processes, and documentation; software process maturity scale	C
Algorithmic strategies (2 hours): brute-force; greedy; branch-and-bound; heuristics; pattern matching; string/text	C
Basic algorithmic analysis (3 hours): asymptotic analysis of upper and average complexity bounds; best, average, and worst case behaviors; big O and little o notations; standard complexity classes; empirical measurements of performance; time and space tradeoffs; recurrence relations	C
Concurrency (2 hours): threads; scheduling, synchronization and timing; multi-threaded programs; race conditions	C
Professionalism (1 hour): standards of professional behavior; professional computing societies and publications; professional responsibilities and liabilities; ACM Code of Conduct; career paths in computing	C

Introduction to Computer Security

Topic	Bloom's Level
Security goals and fundamentals: confidentiality, integrity, availability, reliability, etc.	K
Secure systems: types, models, design, changes to non-secure systems; comparative analysis	C
Access controls: controlling access to resources, access matrix model, access control lists and capability lists; mandatory controls, originator controls	C
Networks and security: internet security architecture, internet protocols, implementation considerations; firewalls	C
Integrity: cryptographic checksums, malicious logic, viruses, Trojan horses; defenses, prevention	K
Cryptography fundamentals: classical, public key; implementation problems	K
Authentication: passwords	C
Attacks: software attacks (malicious code, buffer overflows, social engineering, injection attacks, and related defense tools); network attacks (denial of service, flooding, sniffing and traffic redirection, defense tools and strategies); website attacks (cross-site scripting)	K
Management: planning for security; introduction to risk assessment and management; business cases; regulatory compliance and legal issues; Federal Information Security Management Act; and business continuity/disaster planning	K
Security standards in government and industry: NIST 800-39 (risk management), NIST 800-53 (security controls), ISO 27001, and ISO 27002; sample corporate and institutional security policies	K
Security issues in requirements, architecture, design, implementation, testing, operation, maintenance, acquisition, and services	K
Ethics and professionalism as related to computer security	K

Secure Coding

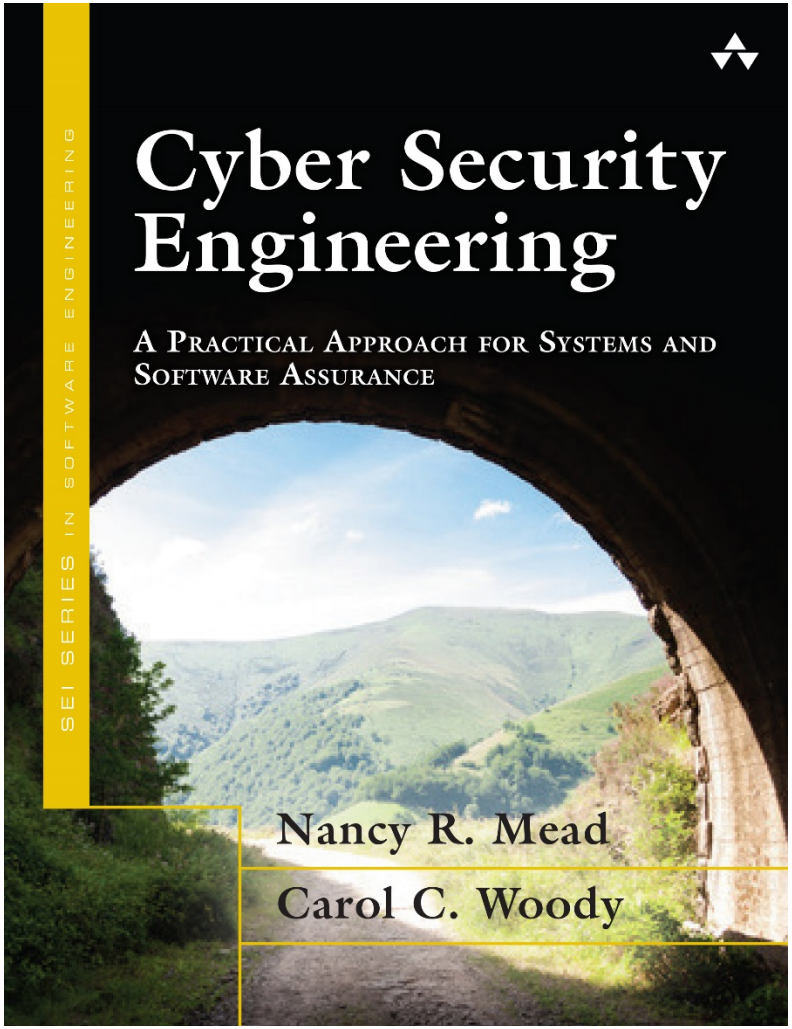
Topic	Bloom's Level
Overview of security vulnerabilities and risks in software: Common Weakness Enumeration (CWE), Open Web Application Security Project (OWASP) Top 10	C
Data protection: methods for preventing unauthorized access or manipulation of data	AP
Input validation and user authentication	AP
Memory management: buffer overflow, memory corruption, and privilege violations	AP
Integer overflow and misuse of strings and pointers	AP
Communication vulnerabilities: concurrency, secure inter-process communication and authorization, authentication and networking protocols	AP
Unit testing for security vulnerabilities: fuzzing, abuse cases	AP
Code review: formal inspections and static analysis	AP
Vulnerabilities in modern languages: insecurities in Java and hypertext preprocessor (PHP)	C
Standard risk mitigation strategies and resources: coding standards, enterprise security API (ESAPI)	C
Professional development: OWASP, certification	C

Introduction to Assured Software Engineering

Topic	Bloom's Level
Introduction to software project management: project planning, estimation, configuration management, risk management; and software security process models: Building Security In Maturity Model (BSIMM), OWASP Software Assurance Maturity Model (SAMM), Microsoft Software Development Lifecycle (SDL)	C
Role of assured software engineering: software engineering for assurance and its place as an engineering discipline	C
Requirements analysis: requirements analysis for functional and quality requirements	AP
Introduction to software architecture: introduction to software architecture, including architectural patterns (pipe & filter, MVC), client-server computing	C
Use and misuse cases: use cases, misuse cases, and user-centered design	C
Design patterns: abstraction-occurrence, composite, player-role, singleton, observer, delegation, facade, adapter, etc.	C
UML: Review of object-oriented principles, UML class diagrams, and object-oriented analysis	AP
Domain modeling: examples of building class diagrams to model various domains	C
Reusable technologies: Review of reusable technologies as a basis for software engineering, risks associated with reuse (e.g. Ariane)	C
Software behavior: representing software behavior: sequence diagrams, state machines, activity diagrams, correctness under all conditions of use	AP
Verification and validation: Inspections and reviews, integration, system, and acceptance testing	AP

Additional Resources

Publication to Support the Curriculum



Released November 2016 as part of the SEI Book Series

For more information see https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=483667

CERT Cybersecurity Engineering and Software Assurance Professional Certificate

Released March 2018

The program consists of five components

- Software Assurance Methods in Support of Cybersecurity Engineering
- Security Quality Requirements (SQUARE)
- Security Risk Analysis (SERA)
- Supply Chain Risk Management
- Advanced Threat Modeling

To learn more, visit https://sei.cmu.edu/education-outreach/credentials/credential.cfm?customel_datapageid_14047=33881.

Contact Information

Carol Woody
cwoody@cert.org

Web Resources (CERT/SEI)
www.sei.cmu.edu/go/cybersecurity-engineering