Institute of Information Technology
University of Dhaka

Course Curriculum

Course Code:	
Course Name:	Secure Software Design
Course Credit:	3 (Theory 2 + Lab 1)

Designed by: Dr. Mohammed Shafiul Alam Khan

Course Instructor: Dr. Mohammed Shafiul Alam Khan

Course Description

Objective:

This course will introduce the theoretical concepts and practical approaches and tools that support the security concerns in the complete systems development lifecycle resulting in software that is secure by default. The course will cover a wide range of software security topics ranging from as security as a cross-cutting concern, methodological approaches to improving software security during different phases of software development lifecycle, integrating secure software development principles and patterns into software development processes.

Outcome:

In this course, students will be exposed to the techniques needed for the practice of effective software security techniques. By the end of the course, one should be able to do the following things:

- Security risk management. Students will be able to assess the security risk of a system under development. Risk management will include the development of formal and informal abuse case and threat models. Risk management will also involve the utilization of security metrics.
- Security testing. Students will be able to perform diffeent types of security testing, including fuzz testing at each of these levels: white box, grey box, and black box/penetration testing.
- Secure coding techniques. Students will understand secure coding practices to prevent common vulnerabilities from being injected into software.
- Security requirements, validation and verification. Students will be able to write security requirements (which include privacy requirements). They will be able to validate these requirements and to perform additional verification practices of static analysis and security inspection.
- Vulnerability Assessment. Students will be able to evaluate a medium-sized application for vulnerabilities and generate a report on the evaluation and its results.

Textbook:

- Software Security: Building Security In, -- Gary McGraw, ISBN: 0-321-35670-5
- Exploiting Software How to Break Code -- Greg Hoglund, Gary McGraw, Publisher: Addison Wesley, ISBN: 0-201-78695-8
- Writing Secure Code, 2nd edition. Microsoft Press, Redmond, WA, 2003. Michael Howard and David LeBlanc

Reference Resources:

- Software Security Principles, Policies, and Protection -- Mathias Payer April 2019, v0.35
- Online resources and lecture notes provided during class

Theory: 28 hours

SL#	Topics	Coverage (hours)
1.	Introduction to Software Security: why software security is a concern (the trinity of trouble), security problems in software (bugs, flaws, defects, vulnerability and exploits), introducing three pillars of software security (applied risk management, software security touchpoints, and security knowledge)	1.5
2.	Software Risk Management Framework (RMF) [different activities in the RMF]	1.5
3.	Introduction to software security touch points and its relationship with software development life cycle / security development process	1.5
4.	Knowledge for software security: Perspective Knowledge [Security principle, Security guideline, Security Rules], diagnostic knowledge [attack patterns, exploits, and vulnerability], and historical knowledge	1.5
5.	Software security principles: SD^3 (Secure by Design, by Default, and in Deployment), Minimize Your Attack Surface, Use Defense in Depth, Use Least Privilege, Backward Compatibility Will Always Give You Grief, Never Depend on Security Through Obscurity Alone, Don't Mix Code and Data, Remember That Security Features! = Secure Features	2
6.	The Proactive Security Development Process [process improvement, design, development, testing and maintenance]	1.5
7.	Secure design through threat modeling and different security techniques	2.0
8.	Secure Coding Techniques: The buffer overrun	1.5
9.	Secure Coding Techniques: Determining Appropriate Access Control	1.5
10.	Secure Coding Techniques: Running with Least Privilege	1.5

11.	Secure Coding Techniques: Appropriate use of cryptography	1.5
12.	Secure Coding Techniques: Protecting Secret Data	1.5
13.	Secure Coding Techniques: All inputs are Evil! So, beware of your inputs	1.5
14.	Security Testing: The role of security testing, Building Security Test Plans from a Threat Model (risk-based security testing), developing abuse cases, Determining Attack Surface, Testing for negatives (penetration testing)	3
15.	Code review and taxonomy of coding errors	1.5
16.	Case Study 1: Web Security [browser security, command injection, SQL injection, XSS, and XSRF]	1.5
17.	Case Study 2: Mobile Software Security [android system security, android permission model, reverse engineering and program understanding]	1.5
	Total	**28 hours**

Lab: 28 hours

SL#	Lab	Coverage (hours)
1.	Use of static analysis tools for security testing (source code analysis): Fortify / ITS4	6
2.	Buffer Overflow vulnerability lab	4
3.	Cross-Site Request Forgery attack lab	4
4.	Cross-Site Scripting attack lab	3
5.	SQL injection attack lab	3
6.	Android repackaging attack lab	4
7.	Online application security testing: OWSAP ZAP / Acunetix	4
	Total	**28 hours**