Course name	Secure Software Development

Course ID:	*40-???*	**Credits:** 3		**Program:**	Graduate
Prerequisites:	Advanced Programming (40-244), Software Engineering (40-474)			**Co-requisites:**	Data & Network Security (40442).
Prepared by:	Rasool Jalili & M.S. Dousti				

1. Aim

The course introduces the secure software development process including designing secure applications, writing secure code which can withstand attacks, and security testing and auditing. It focuses on the security issues a developer faces, common security vulnerabilities and flaws, and security threats. The course explains security principles, strategies, coding techniques, and tools that can help make code more resistant to attacks. Students will write and analyze code that demonstrates specific security development techniques. The course objectives are:

1. Understand the basics of secure programming.
2. Understand the most frequent programming errors leading to software vulnerabilities.
3. Identify and analyze security problems in software.
4. Understand and protect against security threats and software vulnerabilities.
5. Effectively apply their knowledge to the construction of secure software systems.
6. Make effective use of tools to assess software security.
7. Understand the low-level features of the CPU, OS, and software.
8. Ability to reverse engineer a given piece of software.1

2. Outline

The course is divided into 3 sections: software security, reverse engineering, and secure coding.

2.1 Software Security

Software security aims to build security inside software. To provide this target, students will learn the following topics based on [1].

1. Introduction and fundamentals
2. Risk and Risk Management
3. Main steps in Software Security
4. Code Review
5. Architectural Risk Analysis
6. Software penetration Testing
7. Risk-based Security Testing

8. Abuse Cases

9. Software Security and Security Operations

10. Enterprise SW Security Testing

2.2 Reverse Engineering

Reverse engineering is the process of discovering the technological principles of a device, object or system through analysis of its structure, function and operation. In software context, it often involves taking the software apart (using tools such as disassembler, decompiled, debugger, etc.) and analyzing its workings in detail to be used in maintenance, or to try to make a new program that does the same thing without using or simply duplicating (without understanding) any part of the original.

In this section of the course, we look deeply at the low levels of the operating system, as well as the code itself. Following a crash course on the x86 assembly language, the student is introduced to the way an executable is loaded in the memory, the way values are stored in the memory, how a software protects itself against reverse engineers, and so on.

We use [2] as the main reference for this section, but online materials (such as [3, 4, 5]) will also be handy.

2.3 Secure Coding

The section of the course is partitioned into four major subsections:

_ Web Applications security issues

_ Implementation security issues

_ Cryptographic security issues

_ Networking security issues

It tries to cover almost every security issue a developer may face while programming a software, such as “SQL injection”, "buffer-overflow" “XSS," and so on.   Our main reference is [6].

3. Evaluation Criteria

Due to the practical nature of the course, the practical knowledge of the students should be evaluated. Grades will be assigned on a percentage basis for the following areas (tentative):

1. Assignments: 40%
2. Project: 30%
3. Exam: 30%

4. References

[1] Gary McGraw, Software Security: Building Security In. Addison Wesley Professional, 2006.

[2] Eldad Eilam, Reversing: Secrets of Reverse Engineering. Wiley Publishing, Inc., 2005.

[3] http://www.reversing.be/

[4] http://www.tuts4you.com/

[5] http://crackmes.de/

[6] Michael Howard, David LeBlanc, and John Viega, 24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them. McGraw Hill, 2010.