CSC 666–Secure Software Engineering

Spring 2009
MW 7:45-9:00

Instructor Information

Name	:	James Walden
E-Mail	:	waldenj@nku.edu
Office	:	ST 340
Phone	:	(859) 572-5571
Web Site	:	http://www.nku.edu/~waldenj1

	Office Hours
MW	1:00-2:00
TR	10:00-11:00
Fr	by appt

Summary

Description	:	Secure software engineering focuses on creating software that functions correctly even when attacked. Topics include common software vulnerabilities, risk analysis, misuse cases, secure design principles and patterns, secure programming techniques, code reviews, and security testing. Students need to have a basic level of understanding of both software engineering and information security before taking this course.
Prerequisites	:	CSC 540: Software Engineering CSC 582: Computer Security
Textbooks	:	Gary McGraw, *Software Security*, Addison-Wesley, 2006. Brian Chess and Jacob West, *Secure Programming with Static Analysis*, Addison-Wesley, 2007.

Student Learning Outcomes

By the end of the course, a successful student should be able to

1. Explain the nature and importance of software security.
2. Explain common security vulnerabilities, such as buffer overflows, cross-site scripting, and injection flaws.
3. Explain software security techniques for requirements, design, implementation, and testing of software.
4. Evaluate the security risks of an application, using code reviews and security testing.

Grading

Your grade in this course will be based primarily on a set of web application security assignments. Your grade will also include midterm and final examinations.

		A	→	90	–	100
midterm	20%	B	→	80	–	89
final	20%	C	→	70	–	79
assignments	60%	D	→	60	–	69
		F	→	0	–	59

Students with Disabilities

Students with disabilities who require accomodations (Academic adjustments, auxiliary aids or services) for this course must register with the Disability Services Office. Please contact the Disability Service Office immediately in the University Center, Suite 320 or call 859-572-6373 for more information. Verification of your disability is required in the Disability Services Office for you to receive reasonable academic accomodations. Visit our website at `http://www.nku.edu/~disability/`.

Academic Dishonesty

The work that you submit in this course is subject to Northern Kentucky University's Student Honor Code (see `http://www.nku.edu/~deanstudents/documents/StudentHonorCode-Fall2007.pdf`.) Issues involving academic dishonesty are taken very seriously by this instructor and are dealt with according to College and Department policy. Academic dishonesty includes but is not limited to:

1. Improper access to evaluation material or records.
2. Submission of material which is not the student's own work.
3. Conduct which interferes with the work or evaluation of other students.

Some specific examples of dishonesty include:

1. Copying from another person, book, magazine, or other electronic or printed media.
2. Obtaining another person's exam answer or answers.
3. Assisting another student in submitting work that is not the student's own.

It is unacceptable to share program code. It is unacceptable to share homework solutions. It is acceptable and often a good idea to talk about program algorithms and homework solution strategies, but it is not acceptable to use the same code or code segments, or to share actual solutions to homework problems. Any act of academic dishonesty will result in a grade of zero (0) for that item for the first occurrence. An automatic F in the course will result for the second offense. This policy holds for homework assignments and programs, as well as for tests. In order to be fair, penalties will be applied to all parties involved regardless of culpability or fault.

Course Calendar and Class Structure

See the course web site, `http://faculty.cs.nku.edu/~waldenj/classes/2009/spring/csc666/` for a current course schedule.

Class Structure

The course consists of the following modules:

1. **What is Software Security?**
   The security problem. Security Software $\neq$ Software Security. Bugs vs. flaws. An example bug: SQL Injection.

2. **Code Reviews and Static Analysis**
   The code review process: preparing for a code review, roles and responsibilities, running a code review meeting, remedying defects discovered during the review process, limitations of manual reviews, using a static analysis tool to automate code reviews, code review metrics.

3. **Threats and Vulnerabilities**
   Case studies of software security exploits will be examined to determine the nature of both the threat and the software vulnerability involved, as well as how the attack exploited the vulnerability. Terminology: threat, risk, vulnerability, attack, and exploit. Taxonomies of coding flaws: seven pernicious kingdoms, OWASP top 10. Attack patterns.

4. **Security Testing**
   Applying the test plan developed during risk analysis. Automating testing. Penetration testing techniques and tools. Usefulness of penetration testing. Testing metrics.

5. **Secure Programming**
   Common attacks prevented by data validation. Data encoding and special characters. Canonicalization. White list vs. black list approaches to data validation. Validating all sources of input: databases, environment variables, headers, shared libraries, and more. Designing an application for validation. Applying cryptography: selecting algorithms and key sizes, generating random keys, using cryptographic APIs securely.

6. **Risk Analysis**
   Practical risk assessment. Identifying assets, threats, and measuring attack surface. Cigital Risk Management Framework. Microsoft's Threat Modeling approach. Developing a risk-based test plan.

7. **Security Requirements**
   Expanding software requirements from what the software should do to also include what the software **shouldn't** do. Applying attack patterns to generate Misuse Cases.

8. **Secure Design Principles and Patterns**
   Secure design principles: least privilege, fail-safe defaults, separation of privilege, etc. Secure design patterns for web applications, covering topics like authentication, session management, and access control. Designing usable security controls.