Syllabus of ENPM 691: Secure Programming in C

Spring Semester 2018

Instructor: Dharmalingam Ganesan, PhD
Contact: dganesan@umd.edu
Class hours: Thursday 7:00 PM to 9:40 PM
Class location: TBA

Course Description:

This course teaches the fundamentals of secure programming in C. An in depth discussion on various security vulnerabilities (e.g., buffer overflows) in C applications will be taught with hands-on demo of concepts during the class. Students will learn how a C program runs "under-the-hood". The course will teach nitty-gritty of C programs by analyzing at the assembly level. The course discusses best practices (e.g., coding standards) and design principles for secure programming so that security can be built-in during design time. In addition to assignments, students are required to perform a project related to this course.

Course Prerequisite: Equivalent of ENEE 150

Students taking this course should have prior knowledge of C. In particular, this course assumes that the students are familiar with basic C constructs such as control flow, loops, arrays, structures, pointers, and File I/O. If you have not programmed in C but used other similar programming languages, you may talk to the instructor. Some familiarity with the UNIX environment will also be helpful.

The course requires a fair amount of effort to keep up with the pace of the class. It is a highly technical class. Students should be prepared to devote time to gain the most!

Learning Outcome:

- Understand the fundamentals of secure programming.
- Perform security attacks (e.g., buffer overflow, format string vulnerabilities).
- Debug C programs and understand "under the hood" behavior.
- Learn machine/assembly representation of C programs.
- Analyze C programs for security vulnerabilities.

Recommended Reading Materials:

This course will leverage the following resources. Many textbooks will be referenced because this course requires the student to learn fundamentals of computer systems from a programmer's perspective, assembly level programming and debugging. These mandatory skills for secure programming are often not fully described in a single book. Thus, we will cover selected chapters from each of the following books. In addition, we may refer to several online materials (e.g., blogs, presentations, user manuals of Intel IA32/IA64, GNU tools, etc.)

Students need not buy all the following books. The slides and demos of our lectures should be sufficient in general.

- Randy Bryant's and David R. O'Hallaron. *Computer Systems: A Programmer's Perspective*, 2nd Edition.
- Jon Erickson. *Hacking: The Art of Exploitation*, 2nd Edition.
- Robert Seacord. *Secure Coding in C and C++,* 1st Edition.
- K. N. King. *C Programming. A Modern Approach*. W. W. Norton & Company.
- Brian Kernighan and Dennis Ritchie. *The C Programming Language*, 2nd Edition.

Grading:

The tentative final grade breakdown is as follows:

Quiz	10%
Homework	20%
Project	25%
Mid-term	20%
Final Exam	25%

- There will be one quiz, one mid-term, and a final exam. Students will be given at most three homework assignment sheets. In addition, students are expected to submit a project related to this course.
- It is the student's responsibility to inform the instructor of any intended absences for religious observances in advance. Notice should be provided as soon as possible but no later than the end of the schedule adjustment period.
- **Academic Integrity:** The University's Code of Academic Integrity is designed to ensure that the principle of academic honesty is upheld. All students are expected to adhere to this Code. All acts of academic dishonesty will be dealt with in accordance with the provisions of this code. Please visit the following website for more information on the University's Code of Academic Integrity: http://www.studenthonorcouncil.umd.edu/code.html
- **Honor Pledge:** All assignments and exams for this course are governed by the Honor Pledge: "I pledge on my honor that I have not given or received any unauthorized assistance on this exam/assignment."

Grading Method: Absolute Grading

A+: 95 <= score <= 100
A: 90 <= score < 95
A-: 85 <= score < 90

B+: 80 <= score < 85
B: 75 <= score < 80
B-: 70 <= score < 75

C+: 65 <= score < 70
C: 60 <= score < 65
C-: 50 <= score < 60

D: 0 <= score < 50

Tentative Syllabus:

Week	Topics
1	**Motivation for Secure Programming** Software Security – Why? Example Vulnerabilities A Tour of Computer Systems
2	**Foundations - I** Bits and Bytes

	Hexadecimal Notation Addressing and Byte Ordering
3	**Foundations - II** Integer Security Integer Arithmetic Integer Overflow and Security Vulnerabilities
4	**Machine-Level Representation of C programs - I** Tour of Assembly Language
5	**Quiz (45 minutes) and lecture will continue after the quiz**
6	**Machine-Level Representation of C programs - II** Conditional Statements, Switch-Case Statements, Loops
7	**Stack-based Buffer Overflow** Function calls and Stack Layout Representation of buffers at the assembly level Smashing the stack Protecting the stack
8	**Data Pointer and Function Pointer Vulnerabilities** Smashing the stack by exploiting pointers Dynamic memory allocation and security
9	**Advanced Buffer Overflow Attacks** By-passing non-executable stack Jumping to EAX, ESP, and EMP exploits
10	**Mid-term**
11	**Linking** Load-time exploitation Basics of static and dynamic linking
12	**Format String Vulnerabilities** Stack layout of variadic functions Exploit the format string
13	**File I/O Security** Path Traversal Vulnerabilities File I/O Race Conditions
14	**Concurrency** Multithreads Deadlock and vulnerabilities
15	**Final Exam**

Please note that the instructor may refine and/or exclude certain content if deemed necessary, thus, the order of the weekly content might possibly change during the course.

Computing Requirements:

It is assumed that the students will have access to a laptop which can run virtual machines. During the class, the instructor will use Ubuntu Linux VMWare on a 32-bit machine. However, occasionally a 64-bit version will also be used to demonstrate some concepts to show differences to 32-bit representations. We will also provide virtual machines for you during the first weeks of the semester. Students may use the CD that comes with the book of Jon Erickson (see above) for installing/running a Linux version on their machines. Debugging will be based on GNU's GDB. All C programs will be compiled using GNU's C compiler. The instructor will use the AT&T syntax for teaching assembly language representations of C programs. Occasionally the Intel syntax will be used to highlight differences to the AT&T syntax.