Syllabus
ENPM 691: Secure Programming in C
Spring Semester 2016

Instructor: Dharmalingam Ganesan, PhD
Office: Fraunhofer Center, 5825 University Research Court (Suite 1300), College Park
Contact: 240 487 2915, dganesan@umd.edu
Class hours: Thursday 7:00 PM to 9:40 PM
Class location: JMP 2216

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.)


**Grading:**

The tentative final grade breakdown is as follows:

<table>
<thead>
<tr>
<th>Component</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quiz</td>
<td>10%</td>
</tr>
<tr>
<td>Homework</td>
<td>15%</td>
</tr>
<tr>
<td>Project</td>
<td>20%</td>
</tr>
<tr>
<td>Mid-term</td>
<td>25%</td>
</tr>
<tr>
<td>Final Exam</td>
<td>30%</td>
</tr>
</tbody>
</table>

- There will be one quiz, one mid-term, and a final exam. Students will be given three homework assignment sheets. In addition, students are expected to give a group presentation of 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:

<table>
<thead>
<tr>
<th>Week</th>
<th>Topics</th>
</tr>
</thead>
</table>
| 1    | Motivation for Secure Programming  
Software Security – Why?  
Example Vulnerabilities  
A Tour of Computer Systems |
| 2    | Foundations  
Bits and Bytes  
Hexadecimal Notation  
Addressing and Byte Ordering  
Integer Arithmetic  
Integer Overflow and Security Vulnerabilities |
| 3    | Floating Point  
Converting floating point into binary format  
IEEE Standard for floating point representation |
| 4    | Machine-Level Representation of C programs  
Tour of Assembly Language |
| 5    | Quiz (45 minutes) |
| 5    | Stack-based Buffer Overflow  
Function calls and Stack Layout  
Representation of buffers at the assembly level  
Smashing the stack  
Protecting the stack |
| 6    | Data Pointer and Function Pointer Vulnerabilities  
Smashing the stack by exploiting pointers  
Dynamic memory allocation and security |
| 7    | Advanced Buffer Overflow Attacks  
By-passing non-executable stack  
Jumping to EAX, ESP, and EMP exploits |
| 8    | Mid-term |
| 9    | Format String Vulnerabilities  
Stack layout of variadic functions  
Exploit the format string |
<table>
<thead>
<tr>
<th>Week</th>
<th>Topics</th>
</tr>
</thead>
</table>
| 10   | **Linking**  
|      | Static Linking  
|      | Relocatable Object Files  
|      | Dynamic Linking and Shared Libraries  
|      | Security Vulnerabilities and Linking |
| 11   | **Concurrency**  
|      | Multithreads  
|      | Race Condition, Deadlocks, and vulnerabilities |
| 12   | **Signals and Exception Handling**  
|      | Basics of Signals  
|      | Security Vulnerabilities and Signals |
| 13   | **Project Presentations by students** |
| 14   | **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:**

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. 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.