Introduction to software security

**Module Description:** This module discusses common software vulnerabilities, malware and software secure design principles. Software vulnerabilities, such as buffer overflow, integer overflow, are introduced in the first micro module, followed by a discussion of corresponding countermeasures. Virus, worm, Trojan horse and more malware are introduced in the second micro module. The last micro module focuses on software secure design principles and secure coding.

**Prerequisite Knowledge:**  Students are expected to understand the introduction to cybersecurity concepts module and know integer representations, memory layout, and C programming language basics.

**Length of Completion**: This module includes 3 micro modules. The total length of the module is around 13 hours which include 6 hours for labs.

**Level of Instruction:** This module intended for upper division undergraduate students majoring in computer science or computer engineering.

**Learning Setting:** This module is suitable for many forms of delivery: online/in-class/hybrid.

**Lab Environment:**

SEED labs: buffer overflow attack, format string attack

**Activity/Lab Tasks:** There will be in-class discussion and an out-of-class written assignment.

**Lab Files that are Needed:** None

# learning outcomes

MODULE learning oUTCOMES

• Students will be able to describe the common software vulnerabilities, such as buffer overflow, integer overflow, off-by-one error.

* Students will be able discuss the countermeasures of some common vulnerabilities.
* Students will be able to list the unsafe and safe utility functions in C programming language.
* Students will be able to discuss malware, the harm that malware can cause.
* Students will be able to know virus, worm, Trojan horse and how they spread.
* Students will be able to describe some bad coding habits which could lead to problem codes.
* Students will be able to explain the secure software design principles.

# module Details

**Instructional Files and Online Resources that are Needed:**

Slides:

Lesson 1: Failures, flaws, common vulnerabilities (CSP-M3-L1.pptx)

Lesson 2: Malware (CSP-M3-L2.pptx)

Lesson 3: Software security and secure design principles (CSP-M3-L3.pptx)

**Assessment:**

Written homework questions (CSP-HW3.docx)

# lessons

**Overview of Lessons:**

Lesson 1: Failures, flaws, common vulnerabilities

Lesson 2: Malware

Lesson 3: Software security and secure design principles

**Lesson 1 Learning Outcomes:**

Upon completion of this lesson:

* Students will be able to understand the common software vulnerabilities, such as buffer overflow, integer overflow, off-by-one error.
* Students will be able to understand the countermeasures of some common vulnerabilities.

**Lesson 1 Details:**

**Warm Up:**

Pick one or two real world software vulnerability examples, illustrate what problems caused by the vulnerability and which part in the source code lead the problem. For example, the Heartbleed bug is a vulnerability in the OpenSSL cryptographic software library.

**Lesson:**

Topics to be covered in this lesson include:

* Failure, flaw, error, fault, and other concepts.
* Common software vulnerabilities (buffer overflow, integer overflow, off-by-one) and their countermeasures.

**Active Learning Activity:**

Show students a sample memory layout, and discuss what buffer overflow can do to overwrite different memory areas (stack-based buffer overflow, heap-based buffer overflow).

SEED lab: Buffer overflow attack

SEED lab: Format string attack

**Lesson 2 Learning Outcomes:**

Upon completion of this lesson:

* Students will be able to understand malware, the harm that malware can cause.
* Students will be able to know virus, worm, Trojan horse and how they spread.

**Lesson 2 Details:**

**Warm Up:**

Show students some history and analyzes on viruses and/or worms.

**Lesson:**

Topics to be covered in this lesson include:

* Different types of malware
* Virus, worm, Trojan horse, and other malware. How do they spread? How to detect and remove them?

**Active Learning Activity:**

Discussion:

What can we do to prevent from getting infected by viruses?

What can we do to lower the possibility of getting a Trojan house into our computer systems?

**Lesson 3 Learning Outcomes:**

Upon completion of this lesson:

* Students will be able to understand some bad coding habits which could lead problem codes.
* Students will be able to understand the secure software design principles.
* Students will be able to identify the unsafe utility functions in C programming language and understand the safer versions of those utility functions.

**Lesson 3 Details:**

**Warm Up:**

Show students some bad code examples (such as code which has integer overflow problem) and ask them what results the code/arithmetic operation will generate.

**Lesson:**

Topics to be covered in this lesson include:

* Secure software design principles.
* Unsafe and safe utility functions in C programming language.
* Intentional and non-intentional problems caused by programmers.

**Active Learning Activity:**

After discussing the software vulnerabilities, show students some real codes which have problems in them. Let the students find out the problem lines and discuss what problems may be caused by these codes.

Please attribute Dr. Jim Alves-Foss and Dr. Jia Song, University of Idaho  
  
  
  
Except where otherwise noted, this work is licensed under <https://creativecommons.org/licenses/by-nc-sa/4.0/>  
  
Notwithstanding the non-commercial license terms, non-profit educational institutions are granted a non-exclusive, royalty-free license to adapt and use this material, with attribution.  
  
Creative Commons and the double C in a circle are registered trademarks of Creative commons in the United States and other countries. Third party marks and brands are the property of their respective holders.