How Secure are our Computer Systems Courses?

Majed Almansoori, Jessica Lam, Elias Fang, Kieran Mulligan,
Adalbert Gerald Soosai Raj, Rahul Chatterjee

UC San Diego

Wisconsin University of Wisconsin–Madison
Technology in modern society

- Banks
- Buildings
- Hospitals
- Cars
Technology comes with risks

Computer security is important!!!
Are we training our students on computer security?

Top 20 CS programs in the US (according to US News)

Source: https://www.usnews.com/best-graduate-schools/top-science-schools/computer-science-rankings

Security courses are offered as advanced electives

Students can graduate without taking any security course

Result: Software engineers with no security background!
Is security integrated in CS courses?

Prior work evaluated database textbooks.
- Found plenty of SQL injection bugs.

What about other computer science courses?

We focus on Computer Systems
What is a Computer Systems course?

Computer Systems Course
Focuses on the software part

- Application Programming
  - Graphical Interface
  - Application
  - Libraries
  - Operating System
- System Programming
  - Programming Language
  - Assembler Language
- Instruction Set Architecture - “Machine Language”
  - Processor
  - IO System
- Computer Design
  - Firmware
  - Datapath and Control
  - Logic Design
  - Circuit Design
- Fabrication
  - Semiconductors
  - Materials

Image Source: http://users.ece.northwestern.edu/~kcoloma/ece361/lectures/lec01-introduction.pdf
We ask:
Are computer systems courses taught securely?

**RQ1:** Do students use *unsafe* C/C++ functions in their projects?

**RQ2:** Is computer systems course *taught using unsafe functions*?
Unsafe C/C++ functions lead to vulnerabilities

- Buffer Overflow
- Integer Overflow
- Code Injection
- Format String

Exploit unsafe function (e.g. strcpy)
Example: code snippet with `strcpy()`

```
int main(int argc, char** argv) {
    ... 
    char buffer[20];
    strcpy(buffer, argv[1]);
    ... 
}
```

---

**Source**

Make `argv[1]` larger than 20 bytes to cause buffer overflow.

**Destination**

Controlled by user!

**Buffer overflow ⇒ Control the program flow!**
Popular unsafe functions

Level 2
(Easily Exploitable)

- strcpy
- strcat
- gets
- (v)sprintf
- system

Level 1
(Use with caution)

- atoi
- memcpy
- getopt*
- exec*
- (v)snprintf
- realpath
- popen
Code collection

- We considered top 20 R1 universities in the US (According to US News).
- Collected code from course web pages and GitHub.
- We attributed some of the students’ code to instructors (Details in the paper).
- We found:

  - 567.3 KLOC by Students
  - 193.2 KLOC by Instructors

KLOC = Thousand lines of code
Abundant use of unsafe functions

FlawFinder (Analysis tool)

3,099 Invocations by Students → 60% Level 2

4,238 Invocations by Instructors → 55% Level 2
Most used unsafe functions in the dataset

- atoi: 1,837
- strcpy: 1,704
- strcat: 1,350
- (v)sprintf: 1,053
- memcpy: 727

Level 2
Do students and instructors use similar functions?

Usage counts of unsafe functions for a school

<table>
<thead>
<tr>
<th>func₁</th>
<th>func₂</th>
<th>func₃</th>
<th>....</th>
<th>funcₙ</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>1</td>
<td>0</td>
<td>....</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>5</td>
<td>....</td>
<td>9</td>
</tr>
</tbody>
</table>

\[
\text{Sim}(\vec{S}, \vec{I}) = \frac{\sum_j \vec{S}_j \cdot \vec{I}_j}{\|\vec{S}\| \cdot \|\vec{I}\|}
\]

Found high similarity:
- Most universities scored ≥ 0.5
- Four scored ≥ 0.9
Where did students learn about these functions?

Most of them teach/use unsafe functions !!!

- Instructors Code
- Lecture Notes
- Textbooks
The increased awareness towards `gets()`

9 Invocations only → `fgets()` is getting popular

Replace unsafe functions with their safer alternatives!

```c
a.c: In function `main':
a.c:5:5: warning: 'gets' is deprecated; did you mean 'fgets'? [-Wimplmagic]
    gets(c);
    ^~~~
    fgets
/tmp/ccrDÒMRo.o: In function `main':
a.c:(.text+0x24): warning: the `gets' function is dangerous and should not be used.
```
Is it enough to teach the safe alternatives? NO!

```c
int main(int argc, char* argv[]) {
    char buf[240];
    strncpy(buf, argv[1], strlen(argv[1]) + 1);
    ...
}
```

Also controlled by user!

Make argv[1] larger than 240 bytes to cause buffer overflow.

Buffer overflow ⇒ Control program flow!
More than just teaching safe functions

- Teach safe alternatives
- Update material
- Train Instructors
- Grade code security
How Secure are our Computer Systems Courses?

Collected 760+ KLOC

7,337 invocations of unsafe function

Unsafe examples in course resources

Future directions:

Redesign computer systems course

Integrate security in other required courses

https://majedalmansoori.com  malmansoori2@wisc.edu