madS&P

SIGCSE
Special Interest Group
Computer Science Education

Textbook Underflow:
Insufficient Security Discussions in Textbooks Used for Computer Systems Courses

Majed Almansoori, Jessica Lam, Elias Fang,

Adalbert Gerald Soosai Raj, Rahul Chatterjee

UC San Diego

WISCONSIN
UNIVERSITY OF WISCONSIN-MADISON

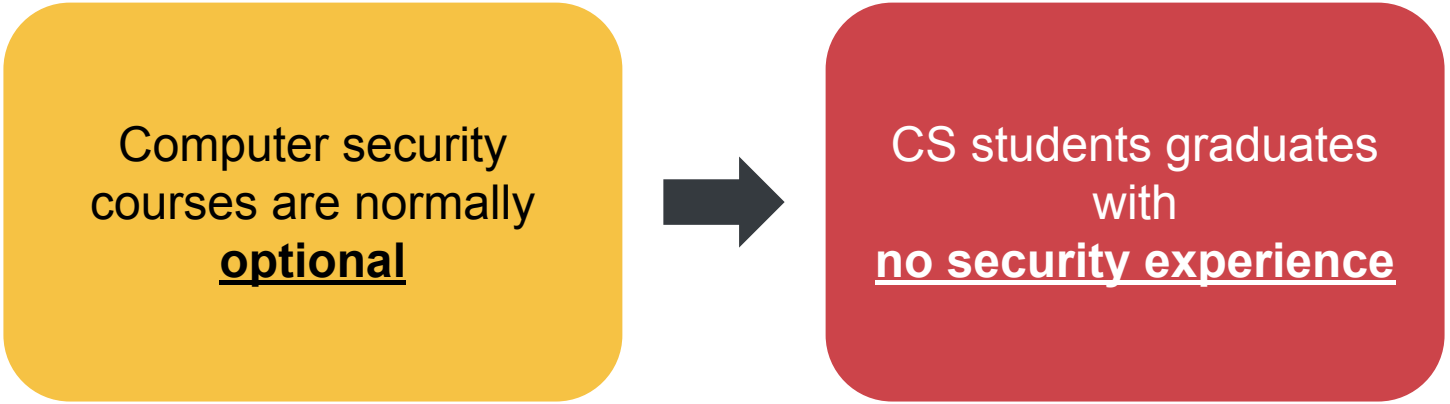
Improving security is crucial

- Reliance on technology has increased.
- Software is developed and updated on daily basis.
- More cyber attacks are seen.

Security experts or trained software engineers?

- Trained software engineers:
  - Performance and user experience.
  - Secure coding habits -- prevents many security flaws
- Security experts:
  - Complex tasks.
  - E.g., testing, mitigation, inspection, etc.

Computer security courses are not required



Security is not integrated in courses

- Plenty of SQL injection queries in Databases textbooks [1].
- Security is not taught in Computer Systems course [2]:
  - *Unsafe C/C++ functions* are used by *students*, and even in *lectures notes*
  - *Security topics* are briefly explained — or not mentioned at all.

[1] Cynthia Taylor and Saheel Sakharkar. '); DROP TABLE textbooks;– An argument for SQL injection coverage in database textbooks. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education, pages 191–197, 2019.

[2] Majed Almansoori, Jessica Lam, Elias Fang, Kieran Mulligan, Adalbert Gerald Soosai Raj, and Rahul Chatterjee. How secure are our computer systems courses? In Proceedings of the 2020 ACM Conference on International Computing Education Research, pages 271–281, 2020.

Unsafe C/C++ functions lead to security issues

- Run shell
- Change flow
- Alter the system
- Reveal information

Exploit
unsafe functions
(e.g. `strcpy`)

List of unsafe functions

Level 2

strcpy
strcat
gets
(v)sprintf
system

Level 1

atoi
memcpy
getopt*
exec*
(v)snprintf
realpath
popen

Example: code snippet with `strcpy()`

```
1  int main(int argc, char** argv) {
2    ...
3    char buffer[20];
4    strcpy(buffer, argv[1]);
5    ...
6  }
```

Source

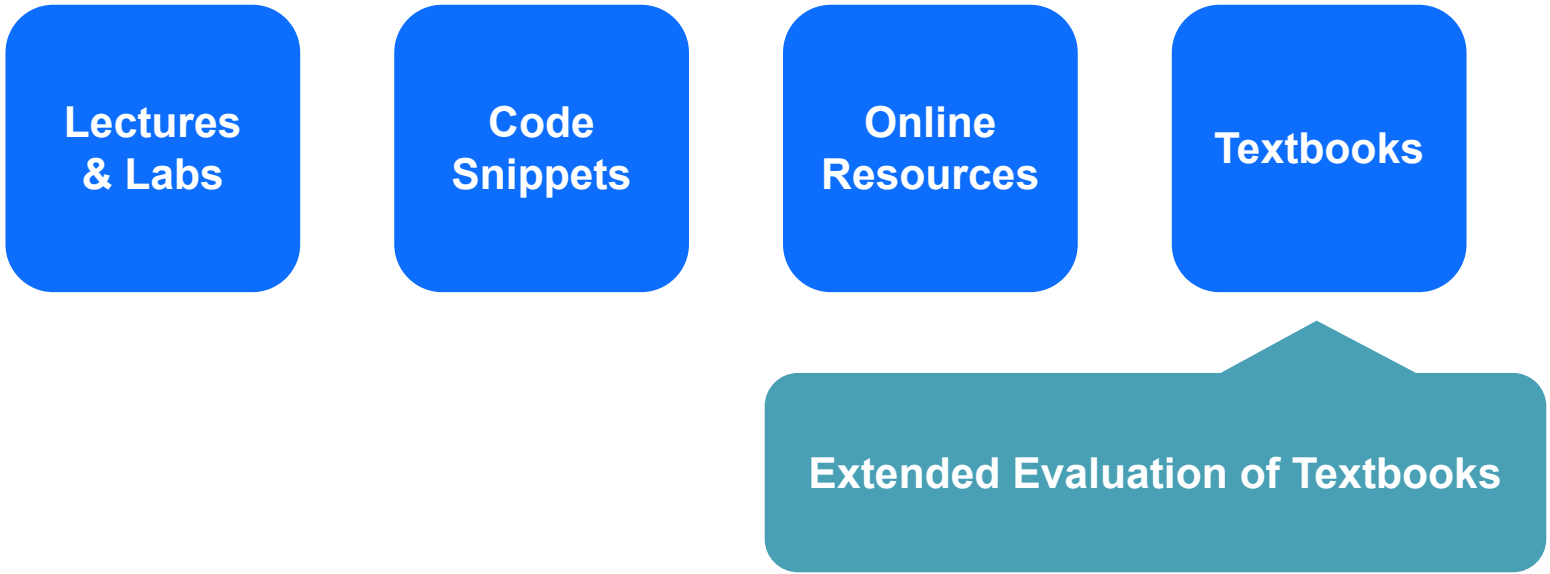
Destination

Controlled by user!

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

Buffer overflow ⇒
Control the program flow!

How do students learn about security?



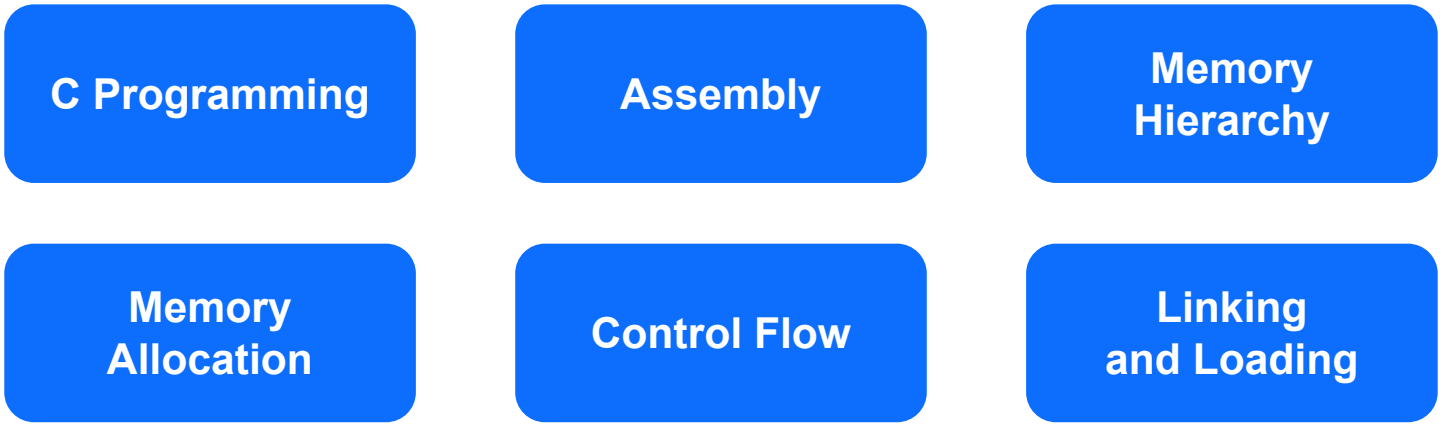
We ask:

RQ1: Are *unsafe functions* used by textbooks in code snippets and are these snippets vulnerable?

RQ2: Do textbooks warn about *unsafe functions* and suggest using *safer alternatives*?

RQ3: How does textbooks discuss *computer security*?

What is a Computer Systems course?



Collecting textbooks

- Picked top **30** undergraduate CS programs in the US (US News Ranking).
  - **5** were excluded -- course is not offered or not taught using C / C++.
- Collected a total of **13** textbooks, including
  - Required & optional textbooks.
  - Different assembly editions (e.g. ARM & x86).
  - All used editions, and the latest editions of the books

Little discussion of unsafe functions in textbooks

- Level 2 unsafe functions:

  `strcpy, strcat, gets, (v)sprintf, system.`

- Key findings: Among 13 textbooks
  - **7** textbooks warned about `gets()`.
  - **No** textbook explained safe usage of `strcpy, strcat,(v)sprintf.`
  - Only **3** textbooks warned about `system`.
  - Only **3** textbooks warned about all unsafe functions.

Vulnerable code snippets in textbooks.

Textbook **kept using** unsafe functions

Few books taught safer alternatives

- Function: `strncpy, strncat, fgets, (v)snprintf.`
- Key findings: Among 13 textbooks
  - **6** textbooks suggested using `fgets()`.
  - **4** textbooks suggested using some **n-version** functions.
  - **3** textbooks did not suggest safer functions at all.
  - Only **1** textbook suggested using safer alternatives for all unsafe functions.

Many textbooks introduce both versions without security explanation !

Incomplete explanation of vulnerable code snippets

Snippet: Multiple Vulnerabilities

```
1  int main(int argc, char** argv) {
2    char command[100];
3    sprintf(command, "ls %s", argv[1]);
4    system(command) // UNSAFE USE OF SYSTEM
5
```

Use snprintf()!!

OS command injection vulnerability!!

sprintf ...

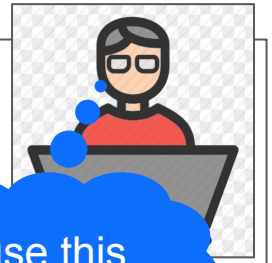
Buffer overflow ⇒ Control the program flow!

Vulnerable code snippets to explain other topics in the book

Snippet: Topics explanation using unsafe functions

```
1  struct food {
2      char name[64];
3      float price;
4  };
5
6  void initFood(struct food f*, char* name, float
7    strcpy(f->name, name);
8    f->price = price;
9  }
```

Could be a user-input!!



I could use this for my project

Use strncpy()!!

Buffer overflow ⇒
Control the program flow!

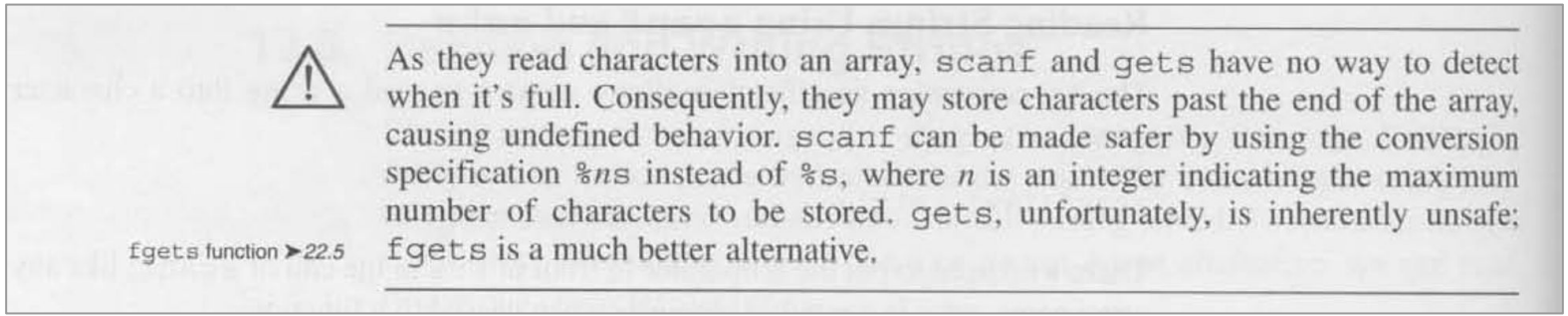
Overflow
f->name

Little discussion of security topics!

- Security topics:
  - Buffer overflow.
  - Integer overflow.
  - OS command injection.
- Key findings:
  - Integer overflow was discussed only as a code performance issue.
  - Buffer overflow is mostly mentioned briefly without a demonstration.
  - Only 3 textbooks discussed OS command injection.
  - Security discussion did not change across different editions.

Security issues are mentioned as “Undefined Behavior”

Sometimes security is explained in terms of performance!



fgets function ➤22.5

As they read characters into an array, `scanf` and `gets` have no way to detect when it's full. Consequently, they may store characters past the end of the array, causing undefined behavior. `scanf` can be made safer by using the conversion specification `%`*n*`s` instead of `%s`, where *n* is an integer indicating the maximum number of characters to be stored. `gets`, unfortunately, is inherently unsafe; `fgets` is a much better alternative.

K. N. King. C Programming: A Modern Approach, Second Edition. W.W. Norton & Company, 2008. Page 286

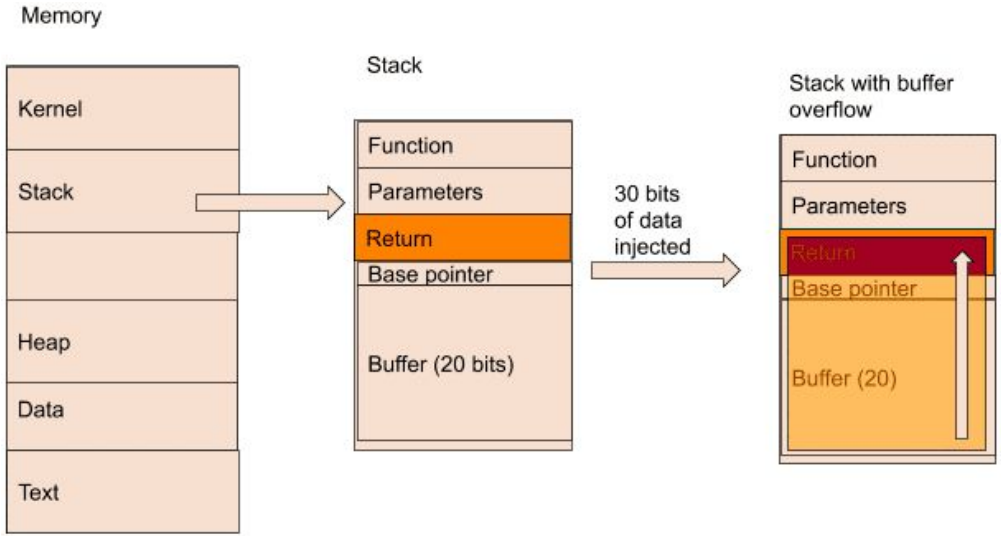
How to improve textbooks?

Never use unsafe functions to explain other topics.

```
1  struct food {
2    char name[64];
3    float price;
4  };
5
6  void initFood(struct food f*, char* name, float price) {
7    // strcpy(f->name, name); !!! NEVER USE IT !!!
8    strncpy(f->name, name, 64); // Prevents Buffer Overflow
9    f->price = price;
10 }
11
```

Only use unsafe functions to demonstrate their issues

Demonstrate how unsafe functions can be exploited



Detailed demonstration of attacks helps generalize security issues

Suggestions

Security is as important as performance!

- Never use unsafe functions to explain other topics.
- Always warn about unsafe functions whenever they are used.
- Demonstrate how unsafe functions can be exploited.
- Explain the correct way of using safer alternatives!
  - They could also be exploited!

Explain the correct way of using safer alternatives!

```
1  int main(int argc, char* argv[]) {
2    char buf[240];
3    strncpy(buf, argv[1], strlen(argv[1]) + 1);
4    ...
5  }
```

Also controlled by user!

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

Buffer overflow ⇒ Control program flow!

Are textbooks used in the course secure?

Analyzed 13 textbooks

Lots of vulnerable code snippets!

Minimal focus of security!

Future directions:

Evaluate students' understanding of security

Integrate security in required courses

https://majedalmansoori.com

malmansoori2@wisc.edu