Textbook Underflow

Insufficient Security Discussions in Textbooks Used for Computer Systems Courses

Majed Almansoori[1], Jessica Lam[2], Elias Fang[2]
Adalbert Gerald Soosai Raj[2], Rahul Chatterjee[1]

[1] University of Wisconsin - Madison [2] University of California, San Diego

ABSTRACT

Introductory computer science courses, such as Computer Systems, could be used to provide the first exposure to computer security to students. However, prior work has shown that, in the US's top R1 universities, computer systems courses are not taught with security in mind. It was also shown that students and instructors use unsafe functions in their code, leading to security vulnerabilities. In this paper, we focused on the textbooks used for computer systems courses. We analyzed the discussion of security topics and the use of unsafe functions in the thirteen textbooks used in the top 30 R1 universities in the US for teaching computer systems. We show that many textbooks do not discuss security at all, while some limit their discussion to "undefined behavior", ignoring that opportunity to discuss potential security issues associated with the undefined behavior. Furthermore, textbooks that talk about security continue using unsafe functions throughout (though not necessarily in vulnerable ways but also without any warning or explanation). We also show that many textbooks do not warn about unsafe functions they use or teach how to use them safely.

CCS CONCEPTS

• **General and reference → Evaluation**; • **Social and professional topics → Computer science education**; • **Security and privacy → Vulnerability management**.

KEYWORDS

Computer security education; Computer systems; Unsafe functions; Textbooks; Buffer overflow; Security vulnerabilities; C and C++

ACM Reference Format:
Majed Almansoori, et al.. 2021. Textbook Underflow: Insufficient Security Discussions in Textbooks Used for Computer Systems Courses. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21), March 17–20, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3408877.3432416

1 INTRODUCTION

Increased reliance on digital technologies in our daily lives warrants more security awareness for our software developer workforce. However, to date, in many universities in the United States, computer security courses are optional, and a student can graduate with a bachelors degree in computer science or related field without taking any computer security course [5]. Therefore, researchers have proposed using core and introductory courses to educate students on basic computer security [8, 18, 19]. Furthermore, introducing students to computer security early on in their bachelor's studies in computer science can help build a *security mindset* [8].

Despite the dire need, our introductory computer science courses are still far from being taught with security in mind [36]. As prior work [5, 41] has shown, instructors, students, and even textbooks pay little attention to security. Instructors regularly use unsafe functions (such as strcpy, strcat, etc.) while teaching and students use those unsafe functions in a vulnerable manner without realizing their security implications [5].

Textbooks have a huge impact on the instruction materials and students' learning [11, 13, 39]. Taylor et al. [41] showed that major database textbooks contain several SQL injection (SQLi) vulnerabilities and have a minimal discussion about security. Thus, a natural question arises, do textbooks in other introductory CS courses also have such serious vulnerabilities? In this work, we consider the textbooks used in computer systems courses in the top R1 universities in the US [30]. Computer systems is a mid-level required course in almost all universities in the US. This course discusses how a single process runs inside a computer and teaches about process memory layout. It is also normally taught in C/C++ and an assembly language. We believe this course is a great place to introduce students to computer security. Furthermore, as textbooks have a huge impact on instructors' lecture content [13] and students' learning, it is important to understand whether or not computer systems textbooks are written with *security in mind*.

Specifically, we aim to answer the following research questions:

RQ1: Do textbooks used in computer systems courses use unsafe functions such as `strcpy` in their code snippets, and are these code snippets vulnerable?
RQ2: Do textbooks warn students about the potential security risk of these functions and introduce them to the safer alternatives (such as `strncpy`)?
RQ3: How do these textbooks discuss topics on computer security?

We considered the top 30 R1 universities in the US that award a bachelor's degree in computer science. We exclude five universities that do not have a computer systems course taught in C/C++. From the remaining 25 universities, we collected the names of 13 textbooks that are used for teaching the courses and obtained their electronic copies. We then consider the set of level two (`L2`) unsafe functions identified by Almansoori et al. in [5], which are `strcpy`, `strcat`, `(v)sprintf`, `gets`, and `systems`. We searched for the use

SIGCSE '21, March 17–20, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8062-1/21/03…$15.00
https://doi.org/10.1145/3408877.3432416

of these L2 functions in the textbooks and manually checked those usages for security vulnerabilities. Next, we looked at the security discussions or warnings associated with those functions in these textbooks. Finally, we looked at whether or not the textbooks discuss the safe ways to use these L2 functions and/or introduces students to the safer alternatives to those functions, such as `strncpy`, `strncat`, `(v)snprintf`, and `fgets`.

Through our analysis, we found the rather unsatisfying status of currently used textbooks in computer systems courses. Many textbooks do not discuss security at all, and the ones that do discuss, often do so only at a cursory level. Only four textbooks provide some level of warning that using unsafe functions might cause a security vulnerability, and even fewer books introduce safer alternatives to unsafe functions (e.g., strncpy instead of strcpy). Moreover, none of the textbooks explicitly explained how unsafe functions could be used safely while continuing to use them (albeit in a mix of secure and insecure ways). Another pattern that emerged from our analysis is that often the issues with unsafe functions are presented as "undefined behavior", ignoring the opportunity to introduce the relevant security concepts. In the end, we found only one textbook that puts considerable effort into explaining unsafe functions' security implications. Thus, we finish with some recommendations about how these textbooks could be improved so that students who use them get sufficient exposure to security and secure coding.

The key contribution of our work is that it adds to the small but growing body of literature that analyzes security considerations in computer science textbooks and raises awareness about the need for immediate changes in the way we as a community write computer science books. More specifically,

- We do a security analysis of the code snippets given in textbooks used in computer systems courses and show that many textbooks contain vulnerable code or insecure invocation of unsafe functions.
- We also show that textbooks rarely talk about security implications of unsafe functions, and only a few introduce students to safer alternatives to these unsafe functions.

2 BACKGROUND ON UNSAFE FUNCTIONS

Almansoori et al. [5] first looked into the use of unsafe functions in code written by students or instructors in computer systems courses in the US. They divide unsafe functions into categories L1 and L2, based on their severity, possibility of being misused, and the availability of safer alternatives. Level two unsafe functions are the ones that can easily lead to security vulnerabilities. Consider the following code snippet:

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

Typically, this program will crash due to the segmentation fault exception if the first command-line argument is longer than 20. (We focus on this since students are typically concerned about program crashing in their introductory computer science courses while learning C/C++). However, this code is also vulnerable to buffer overflow attack [1, 2]: An attacker can pass a specially crafted string (longer than 20 characters) in their first command-line argument and change the program execution flow to execute any arbitrary command of their choice. Buffer overflow has been the reason behind several security vulnerabilities, including some high-profile ones, such as [29, 31]. Most of the iOS jailbreaks are done by exploiting buffer overflow vulnerabilities [10].

It is well known that some C standard library functions, such as `gets`, `strcpy`, `strcat`, and `(v)sprintf`, are prone to be misused, resulting in buffer overflow vulnerabilities. These are referred to as the L2 unsafe functions in [5]. All these functions write to a destination buffer without checking its bounds.

Almansoori et al. [5] also considered `system` as an L2 unsafe function. The function `system` spawns a new shell to execute the command provided in the input string. The problem is that if that string is a user input, then an attacker can perform a command injection attack (CWE 78 [3]).

```
int main(int argc, char *argv[]) {
   char buf[1000];
   sprintf(buf, "wc -l %s", argv[1]);
   int ret = system(buf);
}
```

In addition to a buffer overflow vulnerability in the third line, the code also has a command injection vulnerability. For example, the attacker can provide a command-line argument such as `"dummy; cp /etc/shadow /public/"`, that will copy the sensitive password file `/etc/shadow` into a public folder.

How to avoid unsafe functions. Not every invocation of unsafe functions is always vulnerable. In certain instances, their use is unavoidable. For example, for `strcpy` if the source string is constant, then the use of strcpy is safe, for example, `strcpy(buf, "Hello World!")`. This is typically true for all the L2 functions that cause a buffer overflow. It is also true for `system`: if the input string is constant, it cannot cause command injection vulnerability. The intuitive argument is that if an attacker cannot influence the source string — as it is constant and set by the developer — that particular invocation of the unsafe function will not lead to a vulnerability. Of course, this in no way ensures that the whole code is safe. In this paper, we will consider an invocation of an unsafe function safe if the input source string is constant.

Of course, in certain cases we have to use non-constant source string, and for that there safer alternative to most of the functions, such as `fgets`, `strncpy`, `strncat`, `snprintf`. All these functions ensure that the number of bytes written on to the destination is upper bounded by a value given by the developer — not by the user of the program. We can easily fix the aforementioned snippets' security vulnerabilities by replacing the invocation of unsafe functions with their safer alternatives: `strncpy(buf, argv[1], 20);` and `snprintf(buf, 1000, "wc -l %s", argv[1]);` For `system` function, there is no such simple safer alternative available. It is advisable that developers try to use a library function to achieve the same functionalities, if possible, and otherwise use precaution when using this function.

3 RELATED WORK

Prior work [5] looked into the security of the code snippets written by the students or the ones used by instructors in computer systems

Textbook Title	# Uni.	Focus
(**CSPP**) Comp. Systems: A Programmer's Perspective (2nd Ed. [6], 3rd Ed. [7])	14	Computer systems
(**COD**) Comp. Org. and Design: The Hardware/Software Interface (x86 [35], ARM [34])	6	
(**DIS**) Dive Into Systems [27]	2	
(**ICS**) Introduction to Computing Systems: From Bits & Gates to C/(C++) & Beyond (2nd Ed. [32], 3rd Ed. [33])	1	
(**ISP**) Intro. to Systems Prog.: A Hands-on Approach [38]	1	
(**CPL**) The C Programming Language [20]	9	C language
(**CRM**) C: A Reference Manual [15]	2	
(**CPMA**) C Programming: A Modern Approach [22]	1	
(**PIC**) Programming in C (3rd Ed. [23], 4th Ed. [24])	1	
(**HFC**) Head First C [12]	1	
(**DDCA**) Digital Design and Computer Architecture (x86 [16], ARM [17])	2	Architecture
(**APUE**) Adv. Prog. in the UNIX Environment [40]	1	Unix
(**LPI**) The Linux Programming Interface [21]	1	Linux/Unix

Figure 1: Titles of computer systems textbooks and the number of universities using them. Textbooks are grouped into three categories based on their focus area: systems, C language, and other (e.g., architecture, Linux/Unix, etc.)

courses. They also briefly mentioned that the textbooks and other course materials in computer systems courses either do not discuss security at all or discuss it at a very cursory level. They did not go into the details of how textbooks used in computer systems courses discuss security-related topics, if at all. Given that textbooks have a profound impact on students' learning outcomes [37], in this work, we aim at answering one crucial open question: Do textbooks provide guidance on how to write secure code and avoid particular obvious security vulnerabilities.

Taylor and Sakharkar [41] were the first to analyze security considerations in computer science textbooks. They looked at major database textbooks and found that most do not talk about SQL injection (SQLi) vulnerability at all, despite SQLi being in the top ten most prevalent vulnerability for decades [4]. Even shockingly, they found code snippets in some textbooks that are vulnerable to SQLi attacks. One sad take away from their work is that despite decades of research [14, 18, 19, 25, 43] advocating for including security topics in required computer science courses remain very far from reality. The textbooks, at the very minimum, must ensure that they follow good security hygiene [18, 42].

We, therefore, extend the work by considering textbooks used in the computer systems courses. Computer systems is a mid-level course that teaches students about the process layout and how a program runs inside a memory. This provides an ample opportunity for textbooks to introduce security, or at the very least, teach students how *not* to use certain unsafe functions. However, as we show in this work, most textbooks give little attention to security and even have insecure usage of `L2` unsafe functions.

4 COLLECTING SECURITY DISCUSSIONS AND CODE SNIPPETS FROM TEXTBOOKS

We aim to understand the perspective towards computer security in the textbooks used in the computer systems courses in the top 30 R1 universities in the US. Specifically, we analyze the usage of unsafe functions in the textbooks and how they introduce security-related topics if any.

We consider top 30 R1 universities according to the US News report [30] (which is a superset of universities considered in [5]). We discarded five universities from our study; some don't have an equivalent course to computer systems. Others teach the class with languages other than C/C++, such as Java. We first collected the titles of thirteen textbooks used by these universities from the corresponding course web pages of most recent offerings. Then we collected the electronic copies of the textbooks. If two universities used different editions of the same book, both editions were collected. We also collected the last published edition of these textbooks if available. This will ensure we get a better understanding of the textbook resources that students are exposed to in the computer systems courses. The list of books we consider is given in Figure 1. Some textbooks such as COD [34, 35] and DDCA [17] have two editions for teaching with Intel x86 and with ARM. Finally, we analyze these textbooks to find the usage and discussion of unsafe functions and related security concepts.

For most of the books, we found PDFs where text search is possible. For two books, we only found a scanned copy where text search was not possible. We did not utilize optical character recognition (OCR) because, given the low image quality of the scanned books, the OCR would be erroneous. We, therefore, decided to go over these two books manually.

After collecting the textbooks, we first used the search functionalities provided by PDF readers to narrow down the pages and code snippets for manual analysis. We first looked for any chapters that seemed to discuss software security, particularly buffer overflow. (We focused only on software security for this project, ignoring other security discussions like network security in these books.) For CSPP [6, 7] and DIS [27] books, we found a dedicated section or chapter on computer security, and for LPI [21], we found a chapter on how to write 'secure privileged programs'. For all books where text search was possible, we searched for the following words using a PDF reader: str*cpy, str*cat, *gets, *s*printf, system, buffer overflow, overflow, attack, injection, adversary, exploit, security, smash. For each of the search results, we captured the context where it is being used, and if the text or code snippet seems relevant to answer our security questions, we recorded the whole text and the code snippets with the relevant contexts in a separate spreadsheet.

For the books with scanned copies, we manually went over the entire textbook and took screenshots of security discussions, mentions of unsafe functions, and code examples containing these unsafe functions.

5 RESULTS

We answer our three research questions by analyzing the 13 textbooks we collected. In short, we found that most of the textbooks give little attention to computer security. In the next three subsections, we describe the findings from our research questions.

5.1 Use of Unsafe Functions

Textbooks often use unsafe functions to introduce certain concepts that otherwise will increase cognitive load and will distract from the learning objectives. Nevertheless, using unsafe functions such

Textbook	Unsafe functions				
	`strcpy`	`strcat`	`gets`	`(v)sprintf`	`system`
CSPP	☆!	☆!	★✓	☆!	-
COD	☆	-	-	-	-
DIS	☆!	☆✓	★	☆✓	-
ICS	☆	-	-	-	-
ISP	☆✓	-	-	-	-
CPL	☆!	☆!	☆	☆!	☆
CRM	☆✓	☆	★	☆	☆
CPMA	☆!	☆!	★!	☆✓	☆✓
PIC	☆	☆	☆	☆✓	☆✓
HFC	☆	☆	★	☆!	★!
DDCA	☆!	☆	-	☆	-
APUE	☆!	☆✓	★	☆!	★!
LPI	☆	☆	★	☆	★!

☆ mentions the function in a code snippet or in the text
☆ warns about the function but does not explain how to use it correctly
★ warns about the function and explain how to use it correctly if possible
! contains code snippet(s) with unsafe invocations of the function
✓ used safely in code snippet(s) or only used to warn about its security issue
(-) the function was never mentioned in the book

Figure 2: The figure shows textbooks that discuss the security issue with unsafe functions. Also, invocations of the unsafe functions in these textbooks are marked as safe or not.

Textbook	Safer alternative functions			
	`strncpy`	`strncat`	`fgets`	`(v)snprintf`
CSPP	✗	✗	✓	✗
COD	✗	-	-	-
DIS	✓	✓	✓	✓
ICS	✗	-	-	-
ISP	✗	-	-	-
CPL	✗	✗	✗	✗
CRM	✗	✗	✓	✗
CPMA	✓	✓	✓	✗
PIC	✗	✗	✗	✗
HFC	✗	✗	✓	✗
DDCA	✗	✗	✗	✗
APUE	✗	✗	✓	✓
LPI	✓	✓	✗	✓

✓ introduces the function as a safer alternative to the unsafe version
✗ does not suggest using the safer alternative instead of the unsafe function
(-) the function and its unsafe version were never mentioned in the book

Figure 3: The figure shows how safer alternatives to unsafe functions are mentioned in the textbooks.

as strcpy incorrectly can easily lead to severe vulnerabilities [1, 2], and therefore prior research [5] suggests educators should warn students about using them.

Invocation of unsafe functions. Notably, all books mention at least one unsafe functions (except DDCA [16, 17], which does not have any significant C/C++ code). As shown in Figure 2, The function `strcpy` is used overwhelmingly in all textbooks. The functions `strcat`, `gets`, `(v)sprintf` are equally used by these textbooks. It is remarkable that the function `system` is the least mentioned function by these textbooks, possibly because the function is advanced and more related to operating systems textbooks.

Warning about unsafe function. While the function `strcpy` is the most used function in these textbooks, only four labeled it as unsafe. Among all textbooks, seven warned about at least one unsafe function, and unsurprisingly, the function `gets` was warned the most. However, some books such as CPL [20] and PIC [23, 24] uses `gets` without alerting the reader about its security flaw.

CSPP [6, 7], DIS [27], and LPI [21] are the only textbooks that warned about all unsafe functions that they use — LPI uses and warns about all the five functions including `system`. While a few textbooks warned about unsafe functions, none of them explained how to use them safely. Three textbooks, HFC [12], APUE [40], and LPI [21], explained that allowing the user to pass an argument to the function `system` is dangerous, and thus, the argument should be constant. There is no safe usage of the function `gets`, and therefore it is enough to illustrate that `gets` is not safe and should not be used at all. For the functions `strcpy`, `strcat`, and `(v)sprintf`, no textbook explicitly mentioned that the source string supplied to these functions should be constant. Textbooks only explained that these functions do not check whether the source string fits into the destination buffer. While it can be implied that using constants is safe, textbooks should explicitly mention it when warning about these unsafe functions. LPI [21] briefly suggested using *if-statements* with these functions to prevent overflowing the buffer.

Using unsafe functions even after warning about them. We observe that some textbooks continue using unsafe functions, even after stating them as unsafe. For example, CSPP [6, 7] and APUE [40] used these functions to explain other topics such as child processes and networks. Moreover, several textbooks kept using `strcpy` and `strcat` to explain other topics such as pointers and preprocessors even after stating that they can lead to buffer overflow and crash the program. While most of these usages are safe, some are indeed unsafe, as shown in Figure 2. Even if textbooks used these functions safely, since no textbook explained how to use them correctly, students might copy these invocations to their projects and modify them slightly, making these invocations vulnerable.

Vulnerable code snippets. Though most of the usages of unsafe functions in the textbooks were safe, most textbooks had at least one code snippet with insecure usages of unsafe functions, as shown in Figure 2. Some of the insecure usage are vulnerable. For example, in CPMA [22], page 328, a code snippet uses `gets` while teaching `#define` preprocessor directive in C.

```
#define ECHO(s) (gets(s); puts(s);
...
if (echo_flag) { ECHO(str) }
else { gets(str); }
```

The code uses `gets`, and clearly the resulting code will be vulnerable. Although the learning focus of the code snippet is not `gets`, if a student uses this code snippet in any of their projects, that code will also likely be vulnerable. Textbooks should avoid such usage of unsafe functions.

A similar vulnerable usage we found on page 407 in APUE [40], also illustrated in the left code snippet in Figure 4, which uses `system` function in a vulnerable manner — passes user's input to the system without any filtering. This code is vulnerable to command injection attacks (CWE-78 [3]) where a user can pass any string that will be passed on to a shell for execution. While the goal of the snippet is to introduce students to timing processes; this could have been done using a constant string in the code.

```
 /* Src: APUE [40], page 407 */
int main(int argc, char *argv[]) {
    ...
    do_cmd(argv[i]);
    ...
}


static void do_cmd(char *cmd) {
    ...
    if ((status = system(cmd)) < 0)
    err_sys("system() error");
    ...
}
```

```
 /* Src: CSPP page 734 [6], and
page 749 [7] */
void eval(char *cmdline) {
    char buf[MAXLINE];
    ...
    strcpy(buf, cmdline);
    ...
}
int main() {
    char cmdline[MAXLINE];
    ...
    eval(cmdline);
    ...
}
```

Figure 4: Code snippets showing vulnerable use of unsafe functions in the textbooks. The left code uses gets in an unsafe manner, and the right one uses system with user provided parameters.

On both occasions, the textbooks did not warn the students of the potential security vulnerabilities of these code snippets. Moreover, we also found several code snippets in other textbooks that are not vulnerable as written in the textbook. Still, if any of the functions that invoke unsafe library functions are changed or used in other code snippets, they could easily lead to security vulnerabilities. Therefore, it is not enough to ensure that only the code snippet is safe as a whole, but the textbooks should ensure the usage of functions in the snippets adhere to the high coding standards. We give one such code snippet from CSPP [6, 7] in the right box in Figure 4. While this snippet is safe, any increase in the size of `cmdline` in the function `main` will make the code vulnerable.

5.2 Introduction of safe alternatives

Unsafe functions have safer alternatives that can prevent buffer overflow. These safer functions are: `strncpy`, `strncat`, `fgets`, and `(v)snprintf`. The function `system` does not have an actual alternative, but the `exec` family can usually provide similar functionality more safely. Despite the availability of safe versions of unsafe functions, not all textbooks introduce them as a better version that should be used instead.

As Figure 3 shows, only a few textbooks explicitly advised the reader to use these alternatives. Remarkably, many textbooks suggested using `fgets` instead of `gets`. Moreover, only DIS [27] introduced all four functions as safer counterparts. While some textbooks explained these safer functions, none of them explained that these functions could also be vulnerable if misused, and none of them explained how to use them correctly. Only CPMA [22] book illustrated how to use `strncpy` and `strncat` safely.

5.3 Discussion of Security

The analysis shows that most textbooks do not discuss security at all, as textbooks tend to focus on code performance and functionality rather than security. Only five textbooks discussed security, and only two explained it in detail. Among security discussions, buffer overflow was the most discussed issue in these textbooks. CSPP [6, 7] textbook explained buffer overflow by showing that the function `gets` can corrupt registers and return addresses saved on the stack if the string copied is larger than the buffer. The textbook further explained how to protect against this vulnerability using techniques such as stack randomization and also warned about more unsafe functions such as `strcpy`, `strcat`, and `sprintf`.

Similarly, DIS [27] explained buffer overflow using `scanf`, a function that can also lead to this vulnerability if misused. In this textbook, an example is given on exploiting this vulnerability to change the program's flow. In addition to warning about unsafe functions and explaining protection techniques, the textbook suggested using secure alternatives for some unsafe functions. Other textbooks, such as COD [34, 35] and APUE [40], mentioned buffer overflow briefly; a demonstration of how it exactly works was not included.

HFC [12], APUE [40], and LPI [21] introduced another vulnerabilities related to systems such as command injection. All three textbooks warned about `system` and the potential risk behind it if the user controlled the input. Surprisingly, integer overflow vulnerability was only explained in terms of code performance and was never introduced as a potential security issue. CSPP [6, 7] had a side note about a function with a security vulnerability due to integer overflow. While integer overflow might not be directly a security issue, it can become a security vulnerability in many cases, so it is necessary to warn the reader about it. Other vulnerabilities were mentioned by APUE [40] and LPI [21], but they were not related to systems, rather, mostly related to other topics such as UNIX security. These vulnerabilities were excluded since our primary focus is on security issues related to systems security.

An interesting finding was the term "undefined behavior" used by CPMA [22] and COD [34, 35] to explain specific topics and code behaviors. Code that can crush the program, such as buffer overflow and writing to uninitialized pointers, are described as cases that lead to undefined behavior. In many cases, undefined behavior can lead to security issues, and in other cases, they will break the program. However, the textbook does not explicitly differentiate between these cases and never mentioned that an undefined behavior could be a security hole.

6 DISCUSSION

The repercussions of teaching students with unsafe programming practices in textbooks can be dangerous. We discuss the effect that textbooks can have on students due to using unsafe functions and give recommendations that textbook writers could use to fix the security-related issues.

Textbooks have huge impact on students' learning. Books are considered an important resource in a course, and students' learning experience depends on them [26, 37]. Students learn concepts and code snippets accompanying those concepts from textbooks. Currently, students are learning about unsafe functions from the textbooks and the code snippets in them. They might copy those snippets in their programming assignments. Without proper warning of using unsafe functions, students might use unsafe functions in a vulnerable way. The (insecure) programming practices in the textbooks might also influence students' coding habits that might continue in their professional careers. Therefore, textbooks, especially used in the introductory courses, must use proper security practices.

Additionally, many instructors select specific portions of the textbooks to base their course curriculum (see e.g., [9, 28]). In the case that a textbook includes unsafe functions toward the beginning and does not discuss the security issues of these functions until later in the book, students might miss the portions that cover the unsafe functions and completely miss the security flaws in them. In the worst-case scenario, instructors might not warn students about these unsafe functions as well.

Based on previous works on computer security education, we know that it is crucial for computer science students to gain a basic understanding of computer security (see for example [18, 19]). Therefore, textbooks must follow best security practices and teach about basic security topics. Without secure textbooks, we cannot expect the courses to be taught with sufficient coverage of security.

Recommendations for updating textbooks. A challenge in including security topics in introductory courses is increased cognitive load, which might distract students (and instructors) from the course curriculum's core content. Computer systems courses are organized in a way that ensures students gain a basic understanding of computer organization and systems programming. This course is already heavy in content, and introducing more discussions of security into courses may disrupt the flow of the classes and possibly overwhelm students. Furthermore, adding material on security may affect other course outcomes if the material covered is redundant in future security-related electives. However, we believe augmenting computer systems courses with security contents could be a natural fit given the contents covered.

Based on our observations, it is clear that there are three main issues with textbooks that need to be addressed: (1) security is rarely discussed in most textbooks, (2) there is insufficient warning against using unsafe functions, (3) textbooks sometimes use unsafe functions within their own code snippets (that could be vulnerable if copied incorrectly). To tackle these issues, textbook writers can take some obvious next steps.

1. Use safer alternatives. As shown in our results, most of the textbooks used by computer systems courses describe the behavior of specific unsafe functions but do not explain the dangers of using them or provide safer alternatives. In addition to avoiding using unsafe functions as much as possible, we decided that the best way to combat this security issue in textbooks is to warn students from using unsafe functions and immediately advise them to use safer alternatives instead. By explaining the dangers of unsafe functions and providing ways around them with safer alternatives, students will be able to gain a stronger foundation in secure programming.

2. Warn about unsafe functions as much as possible. Although not all of the textbooks we analyzed had flawed code snippets, most still discussed the use of unsafe functions without any warning or further discussion on how they should be safely used. Some textbooks also warned about the unsafe functions when they were first introduced but continued to use these unsafe functions throughout the book (although in a mix of both safe and unsafe invocations) without an accompanying warning. This poses a problem because, if students are not aware of the possible security flaws in the textbooks they are using, they may use the code snippets or the unsafe functions and cause their code to be vulnerable to security attacks. In addition, textbooks sometimes use unsafe functions to further teach other topics, which can also cause problems down the road.

3. Discuss security often and early. Finally, in our results, we saw that most textbooks did not discuss security at all. And, those that did only addressed it sparsely. Also, it is established that because textbooks are passively teaching security and not explicitly addressing common security issues in computer systems, students are not advised against producing code that is vulnerable to security attacks, like buffer overflow [5]. Thus, discussing security in textbooks early on will ultimately help students be more conscious of possible security issues in their code and how to prevent these security vulnerabilities. It also allows them to practice secure programming principles early on, which will benefit them later in the industry. Finally, referring to security sections often will make it so that students and teachers do not overlook them.

Limitations and future directions. Our analysis sheds light on the security considerations in the textbooks used in computer systems courses. We do not look in-depth at how that actually shapes the security mindset, or the lack thereof, in students who learn from these textbooks.

The textbooks we analyzed are used by the top computer science universities in the US and represent a small sample of textbooks used overall for teaching computer systems or equivalent courses in the world. To solidify our data, we would need to evaluate even more textbooks from other related courses (e.g., networks). The evaluation can also aid in deciding the appropriate fix to the issue.

We should work with authors of textbooks used by computer science courses to provide supplemental resources that cover security and safe programming practices more in-depth for students. We can then study the impact these new textbooks have on students in their programming habits and industry readiness.

7 CONCLUSION

We analyze the discussion of unsafe functions and their usages in code snippets given in the textbooks used in computer systems courses in the top 30 research-intensive universities in the US. We found that textbooks use unsafe functions such as `strcpy` frequently and sometimes in an unsafe manner, without warning students about their security implications or ever explaining how to use them safely. A few textbooks introduce students to safer alternatives to unsafe functions, and only one textbook explained how to use some of these alternatives correctly. We also evaluate the discussion of systems security in these textbooks and found that most textbooks that warn students about potential memory corruption issues with unsafe functions do not present them as security issues, but rather issues of "undefined behavior". Even for textbooks that explicitly discuss some security vulnerabilities, most of these discussions are brief and general. The main goal of this work is to bring awareness to computer science education community to focus on updating textbooks to include more security discussions and avoid teaching students with vulnerable code.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful feedback and suggestions for improvement.

REFERENCES

[1] CWE-121: Stack-based buffer overflow. https://cwe.mitre.org/data/definitions/121.html/.

[2] CWE-122: Heap-based buffer overflow. https://cwe.mitre.org/data/definitions/122.html/.

[3] CWE-78: Improper neutralization of special elements used in an OS command ('OS command injection'). https://cwe.mitre.org/data/definitions/78.html.

[4] Owasp top ten. https://owasp.org/www-project-top-ten/.

[5] 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.

[6] Randal E Bryant, O'Hallaron David Richard, and O'Hallaron David Richard. *Computer Systems: A Programmer's Perspective*. Prentice Hall Upper Saddle River, 2nd edition, 2003.

[7] Randal E Bryant, O'Hallaron David Richard, and O'Hallaron David Richard. *Computer Systems: A Programmer's Perspective*. Pearson India Education Services Pvt. ltd, 3rd edition, 2016.

[8] Justin Cappos and Richard Weiss. Teaching the security mindset with reference monitors. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 523–528, 2014.

[9] Beth Davey. How do classroom teachers use their textbooks? *Journal of Reading*, 31(4):340–345, 1988.

[10] Stefan Esser. Exploiting the ios kernel. *Black Hat USA*, 2011.

[11] Donald J Freeman and Andrew C Porter. Do textbooks dictate the content of mathematics instruction in elementary schools? *American educational research journal*, 26(3):403–421, 1989.

[12] David Griffiths and Dawn Griffiths. *Head First C*. O'Reilly Media, Inc., 2012.

[13] Pam Grossman and Clarissa Thompson. Learning from curriculum materials: Scaffolds for new teachers? *Teaching and teacher education*, 24(8):2014–2026, 2008.

[14] Mario Guimaraes, Meg Murray, and Richard Austin. Incorporating database security courseware into a database security class. In *Proceedings of the 4th annual conference on Information security curriculum development*, pages 1–5, 2007.

[15] Samuel P. Harbison and Guy L. Steele. *C, a Reference Manual*. Pearson, USA, 5th edition, 2002.

[16] David Harris and Sarah Harris. *Digital Design and Computer Architecture, Second Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2012.

[17] Sarah Harris and David Harris. *Digital Design and Computer Architecture: ARM Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2015.

[18] Cynthia E Irvine. What might we mean by "secure code" and how might we teach what we mean? In *19th Conference on Software Engineering Education and Training Workshops (CSEETW'06)*, pages 22–22. IEEE, 2006.

[19] Cynthia E Irvine and Shiu-Kai Chin. Integrating security into the curriculum. *Computer*, 31(12):25–30, 1998.

[20] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice Hall Professional Technical Reference, 2nd edition, 1988.

[21] Michael Kerrisk. *The Linux Programming Interface*. No Starch Press, 2010.

[22] K. N. King. *C Programming: A Modern Approach, Second Edition*. W.W. Norton & Company, 2008.

[23] Stephen G. Kochan. *Programming in C, Third Edition*. Sams, 2004.

[24] Stephen G. Kochan. *Programming in C, Fourth Edition*. Addison-Wesley Professional, 2014.

[25] Lei Li, Kai Qian, Qian Chen, Ragib Hasan, and Guifeng Shao. Developing hands-on labware for emerging database security. In *Proceedings of the 17th Annual Conference on Information Technology Education*, pages 60–64, 2016.

[26] Marlaine E Lockheed, Stephen C Vail, and Bruce Fuller. How textbooks affect achievement in developing countries: Evidence from thailand. *Educational Evaluation and Policy Analysis*, 8(4):379–392, 1986.

[27] Suzanne J. Mathews, Tia Newhall, and Kevin C. Webb. Dive into systems. https://diveintosystems.cs.swarthmore.edu/.

[28] Jeanne Moulton. How do teachers use textbooks and other print materials: A review of the literature. *The Improving Educational Quality Project, South Africa*, 1994.

[29] Satnam Narang. Buffer overflow vulnerability in apple ios and macos devices disclosed. https://www.tenable.com/blog/buffer-overflow-vulnerability-in-apple-ios-and-macos-devices-disclosed, 2018.

[30] U.S. News. Best computer science schools. https://www.usnews.com/best-graduate-schools/top-science-schools/computer-science-rankings, 2018.

[31] Hilarie Orman. The morris worm: A fifteen-year perspective. *IEEE Security & Privacy*, 1(5):35–43, 2003.

[32] Yale N. Patt and Sanjay J. Patel. *Introduction to Computing Systems: From Bits & Gates to C & Beyond*. McGraw-Hill, Inc., USA, 2nd edition, 2003.

[33] Yale N. Patt and Sanjay J. Patel. *Introduction to Computing Systems: From Bits & Gates to C/C++ & Beyond*. McGraw-Hill, Inc., USA, 3 edition, 2019.

[34] David A. Patterson and John L. Hennessy. *Computer Organization and Design, Fifth Edition: The Hardware/Software Interface*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2013.

[35] David A. Patterson and John L. Hennessy. *Computer Organization and Design: The Hardware Software Interface ARM Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2016.

[36] Svetlana Peltsverger and Orlando Karam. Is teaching with security in mind working? In *2010 Information Security Curriculum Development Conference*, pages 15–20, 2010.

[37] Sukma Prasetya. The effect of textbooks on learning outcome viewed from different learning motivation. In *1st International Conference on Education Innovation (ICEI 2017)*. Atlantis Press, 2018.

[38] Gustavo A. Junipero Rodriguez-Rivera and Justin Ennen. Introduction to systems programming: a hands-on approach. https://www.cs.purdue.edu/homes/grr/SystemsProgrammingBook/, 2014.

[39] Mary Kay Stein, Janine Remillard, and Margaret S Smith. How curriculum influences student learning. *Second handbook of research on mathematics teaching and learning*, 1(1):319–370, 2007.

[40] W. Richard Stevens and Stephen A. Rago. *Advanced Programming in the UNIX Environment*. Addison-Wesley Professional, 3rd edition, 2013.

[41] 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.

[42] Michael Whitney, Heather Richter Lipford, Bill Chu, and Jun Zhu. Embedding secure coding instruction into the ide: A field study in an advanced cs course. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 60–65, 2015.

[43] Li Yang. Teaching database security and auditing. In *Proceedings of the 40th ACM technical symposium on Computer science education*, pages 241–245, 2009.