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

Majed Almansoori, Jessica Lam, Elias Fang, Adalbert Gerald Soosai Raj, Rahul Chatterjee
Improving security is crucial

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

Giant Equifax data breach: 143 million people could be affected
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

Computer security courses are normally optional

CS students graduates with no security experience
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.


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()`

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

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

Buffer overflow ⇒ Control the program flow!
How do students learn about security?

- Lectures & Labs
- Code Snippets
- Online Resources
- Textbooks

Extended Evaluation of Textbooks
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?

- C Programming
- Assembly
- Memory Hierarchy
- Memory Allocation
- Control Flow
- Linking and Loading
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

```c
int main(int argc, char** argv) {
    char command[100];
    sprintf(command, "%s", argv[1]);
    system(command) // UNSAFE USE OF SYSTEM
}
```

- **OS command injection vulnerability!!**
- **Buffer overflow ⇒ Control the program flow!**
- **Use snprintf()!!**

- **sprintf...**
Vulnerable code snippets to explain other topics in the book
Snippet: Topics explanation using unsafe functions

```c
struct food {
    char name[64];
    float price;
};

void initFood(struct food f*, char* name, float price) {
    strcpy(f->name, name);
    f->price = price;
}
```

- **Could be a user-input!!**
- **I could use this for my project**
- **Overflow** `f->name`
- **Buffer overflow ⇒ Control the program flow!**
- **Use strncpy()!!**
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!

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 `%ns` 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.
How to improve textbooks?
Never use unsafe functions to explain other topics.

<table>
<thead>
<tr>
<th></th>
<th>struct food {</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>char name[64];</td>
</tr>
<tr>
<td>3</td>
<td>float price;</td>
</tr>
<tr>
<td>4</td>
<td>}</td>
</tr>
<tr>
<td>5</td>
<td>void initFood(struct food f*, char* name, float price) {</td>
</tr>
<tr>
<td>6</td>
<td>// strncpy(f-&gt;name, name); !!! NEVER USE IT !!!</td>
</tr>
<tr>
<td>7</td>
<td>strncpy(f-&gt;name, name, 64); // Prevents Buffer Overflow</td>
</tr>
<tr>
<td>8</td>
<td>f-&gt;price = price;</td>
</tr>
<tr>
<td>9</td>
<td>}</td>
</tr>
</tbody>
</table>

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!

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

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

Buffer overflow ⇒ Control program flow!

Also controlled by user!
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