Supplementary Materials: Towards Finding the Missing Pieces to Teach Secure Programming Skills to Students

Majed Almansoori[1], Jessica Lam[2], Elias Fang[2], Adalbert Gerald Soosai Raj[2], Rahul Chatterjee[1]
[1] University of Wisconsin - Madison,
{malmansoori2, rahul.chatterjee}@wisc.edu,
[2] University of California, San Diego
{jplam,efang,gerald}@ucsd.edu,

A SURVEY QUESTIONS

A.1 Question 1

Version 1. Finish the following code snippet so that it can read a string from the standard input.

```
#include <stdio.h>
int main() {
    char name[8];
    // INSERT CODE HERE
    printf("%s", name);
    return 0;
}
```

A sample run of the program with user input "Hello World" outputs "Hello W". Which of the following lines of code would you replace `// INSERT CODE HERE` with? **(Mark all that apply)**

- `gets(name);`
- `fgets(name, 12, stdin);`
- `scanf("%s", name);`
- `fgets(name, 7, stdin);`

If you have another answer write it down (optional)

Version 2. Finish the following code snippet so that it can read a string from the standard input.

```
#include <stdio.h>
int main() {
    char name[12];
    // INSERT CODE HERE
    printf("%s", name);
    return 0;
}
```

A sample run of the program with user input "Hello World" outputs "Hello World". Replace `// INSERT CODE HERE` with a code that outputs the above mentioned output.

A.2 Question 2

You have the following code snippet

```
int main(int argc, char *argv[]) {
    char buf[28] = “Hello ”;
    // TODO: Add code here
}
```

Write a C code that concatenates a user input from `argv[1]` to `buf`. For example, if `argv[1] = "World!"` then `buf` should be `"Hello World!"`. If needed, you may use any standard library function without worrying about importing the library.

A.3 Question 3

Version 1. Given that the following code snippet includes an error:

```
int main(int argc, char *argv[]) {
    char str1[] = "This code is
                        copying a string.";
    char str2[15];
    strcpy(str2, str1);
    printf("%s\n", str1);
    printf("%s\n", str2);
}
```

Output:

```
pying a string
This code is copying a string.
```

Can you identify and explain the issue? (If you don't know, write "I don't know")

Version 2. Given that the following code snippet includes an error:

```
int main(int argc, char *argv[]) {
    char question1[] = "Do you enjoy
                            coding or gaming?";
    char question2[15];
    strcpy(question2, question1);
    printf("%s\n", question1);
    printf("%s\n", question2);
}
```

Output:

```
ding or gaming?
Do you enjoy coding or gaming?
```

Can you identify and explain the issue? (If you don't know, write "I don't know")

A.4 Question 4

You have the following code snippet:

```
int main(int argc, char *argv[]) {
    char name[15];
    printf("What's your name? ");
    scanf("%s", name);
    printf("Have a nice day %s!\n", name);
}
```

What can go wrong with this code snippet? Explain the issue and how to fix it. (If you don't know, write "I don't know")
Assume that the user input has no spaces. [1]

[1] This line was added later to the question.

A.5 Question 5

Consider the following code snippet:

```
int main(int argc, char *argv[]) {
    char userInput[20];
    sprintf(userInput, "Hello %s", argv[1]);
    printf("%s\n", userInput);
    return 0;
}
```

An example run would be as follows:

```
$ ./a.out "Me"
Hello Me
```

Let `a.out` be the executable file obtained after compiling the above code (Assume '$' is the terminal prompt). Which of the following code executions do you think might crash the program? (The compiler you are using might not crash but other compilers might).

- `$ ./a.out "World!"`
- `$ ./a.out "World! How are you?"`
- `$ ./a.out "0xHack3r_Atk"`
- `$ ./a.out ""`
- All inputs will break the code.
- No input will break the code.
- The second and the forth options only.
- The first and the third options only.

Explain your choice (If you don't know, write "I don't know") [2]

B OBSERVATIONS FROM CODING INTERVIEWS

In this section, we discuss the observations for each theme and how often we see students commit them. The list of main observations and their frequencies are displayed in Figure 1.

Warnings and error messages from compilers and OS. We noted five main observations related to compilers. The first two are ignoring the warning and errors messages thrown by the compiler or the operating system. A number of students noticed that there was a message prompted by the compiler but decided to ignore it anyway and proceed. This is done more for warnings since they do not necessarily break the code than errors, which prevent the code from executing. Although errors cause the process to stop, 4 students ignored errors completely.

Some students, however, tried to read and understand the warning and error messages. For example, two students got errors while testing their code and tried to fix them; however, they failed and decided to ignore the message. On the other hand, two students looked up messages (errors and warnings) they got and tried to learn more about them. Out of all students, only one student stated that they understood the message shown by the compiler and explained it. Three students, on the other end, expressed that they had no clue about those messages.

Coding resources. While answering the survey questionnaire, students were allowed to search online or use documentations. We observed different habits across students when using available resources. Two students did not search online or check documentations to get help despite getting stuck during the interview; rather, they skipped questions or tried to make wild guesses.

Generally, most students looked up answers and documentations at least once; however, we noticed that students are grouped into two main groups: students who read resources carefully and students who just skim and copy results. We found that 15 students did not read search results carefully and just skimmed pages or StackOverflow answers. Moreover, many of these students (a total of 10) who referenced documentations checked examples only and did not spend time understanding how the searched-up function works. Only 5 students spent time reading through documentations and pages carefully. Among all students, we observed that 4 of them copied answers without trying to understand them.

We also found students generally reference the top result or StackOverflow answer without checking other results; 11 students checked only one result or answer, while 2 checked more pages until they found the desired answer. While the top result/answer is sufficient in many cases, we cannot distinguish whether students typically reference more resources or not.

Knowledge of process memory. Few students had enough understanding of process memory layout, and how basic data types are stored in the process memory. Most students have a partial understanding of what happens in the memory due to buffer overflow. Moreover, for five students, memory errors are "scary" / "undefined behavior" of the program and the system. Finally, students cannot

Theme	Observation	#
Compiler and OS messages	- Ignores warnings	4
	- Ignores errors	4
	- Looks up message	5
	- Understands the message	2
	- Distracted by compiler message	1
	- Does not understand compiler's message	8
Resources	- Does not search up when stuck	2
	- Copies answers without understanding	4
	- Reads pages carefully	5
	- Checks top result or answer only	11
	- Checks multiple results or answers	2
	- Checks documentations	19
	- Skims page or documentation	15
	- Checks example uses of functions only	10
Memory	- Has partial or no understanding of memory	4
	- Understands buffer overflow completely	4
	- Has partial or no understanding of overflow	15
	- Mentions undefined behaviour	7
	- Gives no or wrong explanation to the issue	11
	- Fails or struggles to fix overflow	11
Unsafe functions	- Understands vulnerabilities of (1+) functions	5
	- Knows about (1+) safer alternatives	5
	- Does not know (1+) functions are unsafe	16
	- Used at least one unsafe function	17
	- Used safer alternatives for security reasons	5
Security topics	- Understands consequences of long inputs	5
	- Knows about security attacks on buffers	5
	- Avoids writing vulnerable codes	3
	- Writes insecure code	14
	- Suggests wrong mitigation to buffer overflow	11
	- Writes own code instead of safe functions	8

Figure 1: Major observations and the number (#) of students coded with a theme. Here (1+) means one or more.

[2]The last four options were added in version 2

explain what causes certain memory errors, such as "segmentation fault." or "memory corruption detected". A number of students struggle to correct some simple error caused due to buffer overflow. One student explained the memory errors and the cause of buffer overflow in a program. Although we are not expecting students to have detailed knowledge of buffer overflow, but students must know what causes a buffer overflow, and how to fix it.

Unsafe functions. The survey tests students' knowledge about some popular unsafe functions and the downsides of using them. Generally, we mainly observed that students did not know that these functions are unsafe and must be avoided. In most cases, students used at least one unsafe function for their final answers throughout the survey. Very few students understood that unsafe functions are flawed and could overflow the buffer, and even a fewer number of students knew about at least one safer alternative and used them as a solution to avoid buffer overflow.

Security topics and mindset. We observed that few students understood the risks of user inputs being longer than the buffer that tries to read it and that they can cause buffer overflow. Some students stated that buffer overflow could be exploited by hackers to attack programs and computers. Generally, these students wrote secure code that prevents buffer overflow. However, most students had no or limited security knowledge or awareness during coding. For example, several students proposed increasing the size of a buffer as a solution to solve buffer overflow without realizing malicious user behavior. Security mindset [1] is something every CS student must have while writing code.

C EXTEND SOLO TAXONOMY FOR SECURITY KNOWLEDGE

The full taxonomy created to evaluate each theme is given in Figure 2.

REFERENCES

[1] Ambareen Siraj, Nigamanth Sridhar, John A Drew Hamilton Jr, Latifur Khan, Siddharth Kaza, Maanak Gupta, and Sudip Mittal. Is there a security mindset and can it be taught? In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, pages 335–336, 2021.

SOLO level	Pre-structural (P)	Uni-structural (U)	Multi-structural (M)	Relational (R)
Compiler	Recognizes that there is a compiler message but ignores it or does not notice the message at all.	Ignores warnings but focuses on errors and fixes them but not necessarily in a good way.	Always pay attention to the warnings and errors, and fixes them, but does not necessarily understands them.	Explain the reason for the error/warnings, fixes them thoroughly, and understands how they can cause problems e.g. security issues. Spends time to learn and understand about newly encountered warning and errors.
Resources	Knows where to look for resources but doesn't read thoroughly, just copies solution/code	References one source but reads through it carefully to gain a better understanding (i.e. documentation to understand the function and its uses)	Can reference several sources to compare and look for better solutions (i.e. referencing different documentation sources or going to stack overflow and reading several proposed answers)	Can reference several sources and connect the info learned through online resources with prior knowledge and understandings
Memory	Does not know how memory works in context of computer system or have a very vague idea about it.	Knows how a single object or variable is stored in the memory and how to manipulate it but does not understand the interactions of between different objects.	Knows how multiple objects are stored in memory and their relation to each other	Knows how different functions manipulate memory, understands the limitations of memory and how memory manipulation can cause errors/crashes/side effects leading to security vulnerabilities
Unsafe functions	Heard that some functions should not be used but does not know why, and still uses the functions that was warned about	Knows about one unsafe function and avoids using it without understanding the security implication behind it.	Knows about more than one unsafe function and uses their safer alternatives but does not necessarily understand why they are unsafe	Knows about most popular unsafe functions and uses their safer alternatives instead. Understands why they are unsafe and how can attackers exploit them.
Security topics	Heard about security-related issues but does not know their security implications nor understand the error. Unable to fix them correctly	Understands a few errors but not in terms of security. Cannot explain exact consequences of these errors, struggles to avoid them, and refer to the issue as undefined behavior.	Able to fix errors and can avoid them in many cases but is not aware of the security implications behind them.	Able to fix errors efficiently and avoid them using best means. Understands the security implications behind many errors.

Figure 2: Detailed description of each SOLO level for the five themes we considered. The themes are: understanding compiler messages (Compiler), utilization of resources (Resources), knowledge of memory (Memory), knowledge of unsafe functions (Unsafe functions), and understanding of security topics (Security topics).