Introduction

Welcome!

Thank you for your interest in our study. This survey is part of a research study (STUDY2024_00000346) conducted by X and Y at Z University.

Our research aims to improve interpretability of vulnerability reports, enabling developers of all backgrounds to understand and effectively address security vulnerabilities. Please complete the registration form to help us determine your eligibility for participation.

This survey will **take approximately 10-15 min to complete**.

Instructions: Please provide your answers to all questions and complete the tasks to the best of your ability. If you encounter any questions or tasks that you find challenging, feel free to give your best attempt. Your responses will help us ensure that the participants in the later study have the appropriate background in C programming. While some questions are open-ended to allow you to express your thoughts freely, please refrain from providing any personal or personally identifiable information about yourself or others, unless specifically requested.

Contacts: If you have any questions, please contact X.

Demographics

What is your current occupation?

- ○ Undergraduate student
- ○ Graduate student
- ○ Industry developer not working in security
- ○ Industry developer working in security
- ○ [ ] Other

How many years of programming experience do you have?

- ○ Less than 1 year of experience
- ○ 1 to 5 years of experience
- ○ 6 to 10 years of experience
- ○ 11 to 15 years of experience
- ○ More than 15 years of experience

Are you familiar with the C programming language?

- ○ Yes, I am familiar and use it more than one time per week.
- ○ Yes, I am familiar but I use it less than one time per week.
- ○ No, I am not familiar with C.

From "Very Poor" to "Excellent", how would you rank your ability to **understand** C code?

- ○ Very Poor (no understanding of C code; e.g. you encounter a simple C program that prints "Hello, World!" to the screen, but you can't identify what parts of the code correspond to the output or how the program functions)
- ○ Poor (basic understanding but struggle with complex code; e.g. you understand simple C programs like calculating the sum of two numbers or looping through an array but struggle with code that involves pointers, dynamic memory allocation, or data structures like linked lists.)

- Average (understand most code without issues; e.g. you understand C code that includes functions, conditional statements, and loops without difficulty. You can also handle moderately complex code, such as managing arrays of structures or performing file I/O operations.)
- Excellent (understand code well, including complex constructs and debugging; e.g.: you can understand and work with advanced C concepts like multi-threading, intricate pointer manipulations, and complex algorithms. You are also comfortable debugging and optimizing code.)

From "Very Poor" to "Excellent", how would you rank your ability to **write** C code?

- Very Poor (cannot write basic code and often make errors)
- Poor (can write basic code but find complex coding tasks challenging)
- Average (can write code and handle most common tasks without issues)
- Excellent (excel at writing code, including complex scenarios)

Task 1

Consider the following C code snippet:

```
1 int* pc, c;
2 c = 5;
3 pc = &c;
4 c = 1;
5 printf("%d", c);
6 printf("%d", *pc);
```

What is the output of the `printf("%d", c);` statement (line 5) ?

- 1
- 5
- None of the previous

What is the output of the `printf("%d", *pc)` statement (line 6)?

- 1
- 5
- None of the previous

Task 2

Consider the following C code snippet.

```
1 int* ptr = malloc(sizeof(int));
2 *ptr = 42;
3 free(ptr);
4 *ptr = 10;
```

After calling "free(ptr)", the memory pointed to by "ptr" is guaranteed to be deallocated and

can be safely reused for other purposes.

- False
- True

The line "*ptr = 10;" after "free(ptr);" will definitely cause a segmentation fault.

- True
- False

What is a good practice to avoid issues related to using pointers after freeing memory?

- Always set the pointer to NULL after freeing the memory and checking if it is NULL before using it again.
- Never use malloc() without also using free() immediately after.
- Avoid using pointers in your programs.

Block 4

Please, insert **three possible date/times** for the study session before October 20th. Please include your timezone.

We will contact you to confirm the session schedule or propose a different time.
You can expect a message from X.

For contact purposes only, leave your email below.