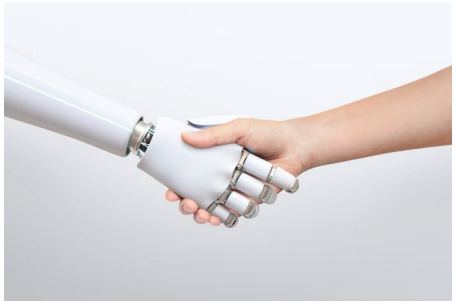
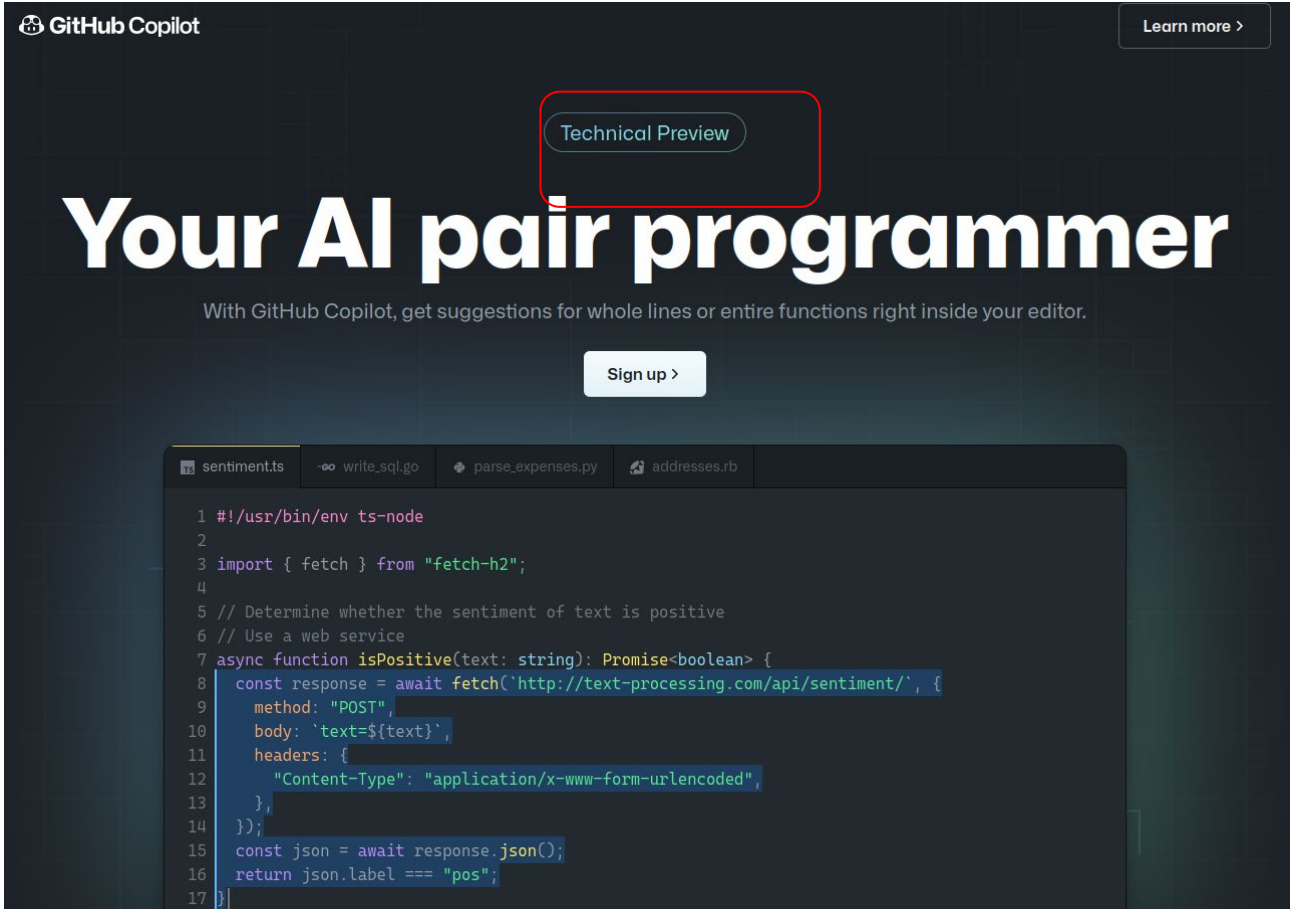
Lost at C: A User Study on the Security Implications of LLM Code Assistants

Gustavo Sandoval, Hammond Pearce, Teo Nys, Ramesh Karri, Siddharth Garg and Brendan Dolan-Gavitt

August 2023



June 29, 2021: Future of software development?



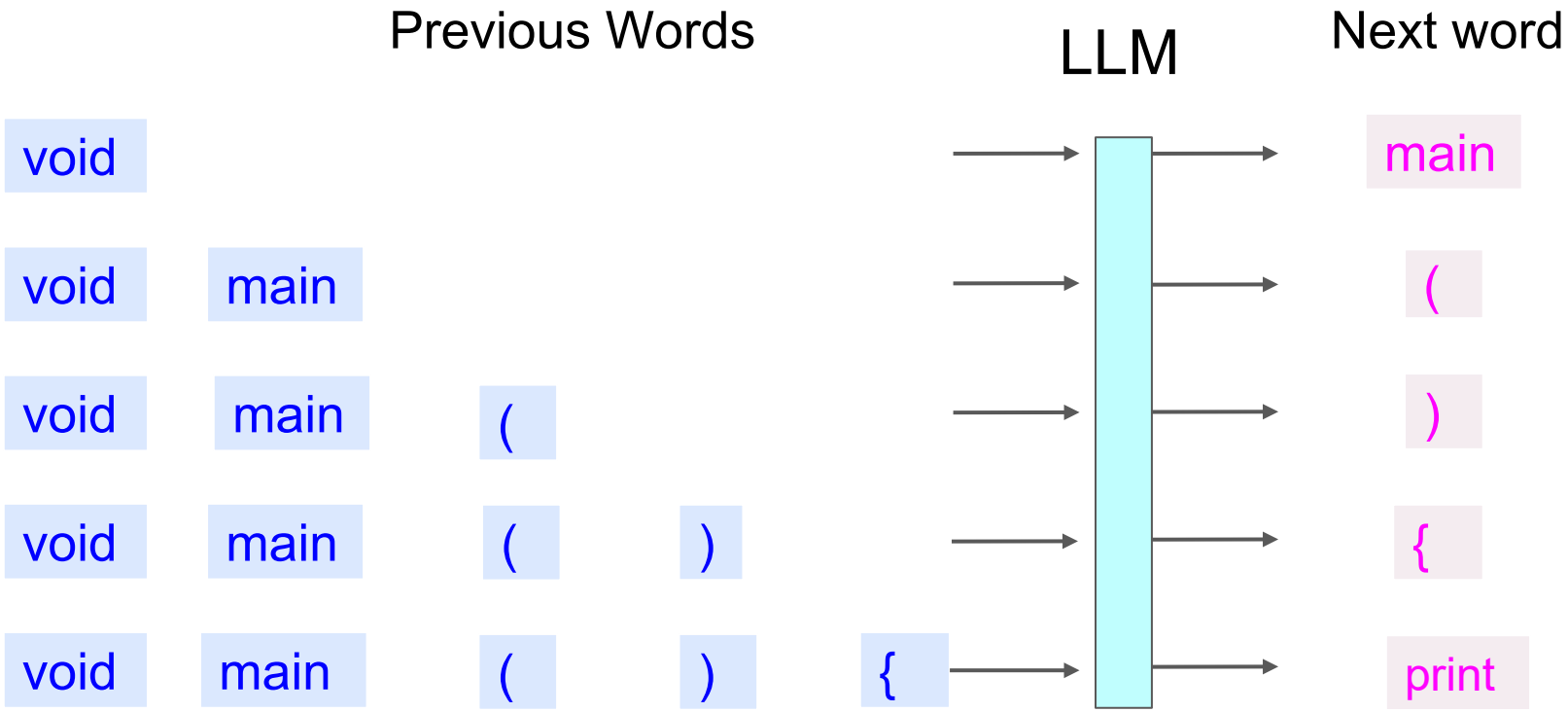
August 2023


Redefining developer productivity.
75% more fulfilled
1M+ developers
46%
code written
55%
faster coding
runtime.go
1 package main
3 type Run struct {
4 Time int // in milliseconds
5 Results string
6 Failed bool
7 }
10 func averageRuntimeInSeconds
5,000+
businesses have used it

What are large language models?

LLMs predict the next word given any sequence of words

What are large language models?



LLMs predict the next word given any sequence of words

Issues with LLM-generated code?

Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions

Hammond Pearce
Department of ECE
New York University
Brooklyn, NY, USA
hammond.pearce@nyu.edu

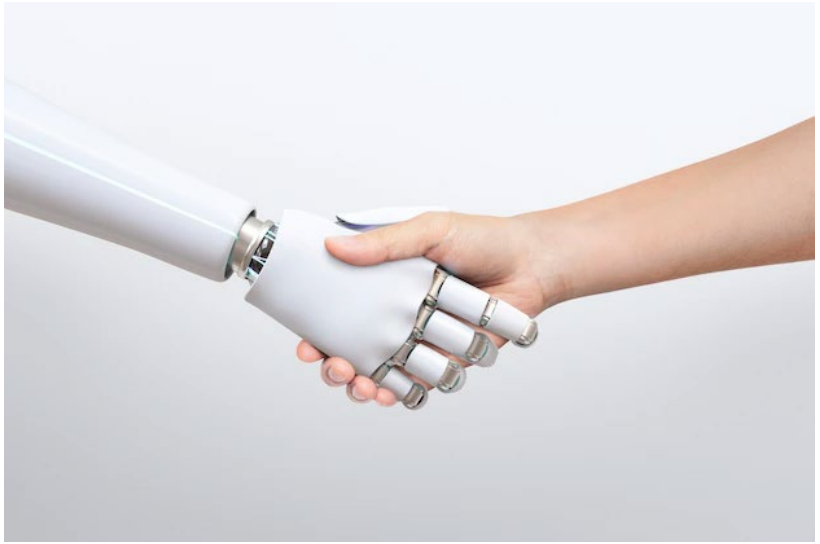
Ramesh Karri
Department of ECE
New York University
Brooklyn, NY, USA
rkarri@nyu.edu

Bugs in 40% of security-related completions

→ Does this impact code written *by users*?

First security user study

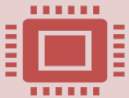
Human (Control)



Human + AI (Assisted)

Research Questions

Does an AI code assistant help novice users write better functional code?

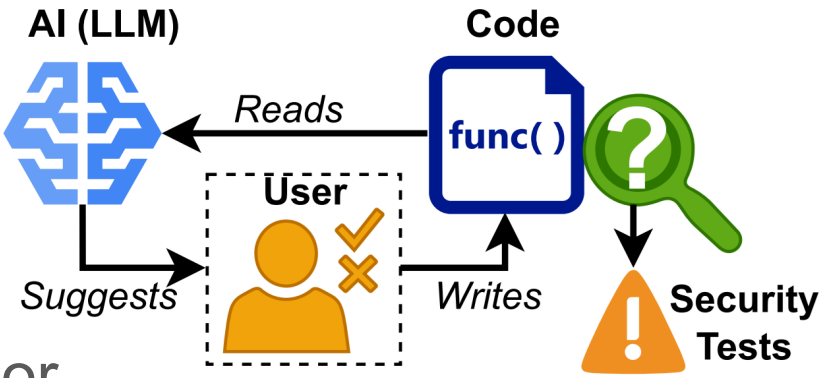


Does code written with AI assistance have an acceptable incidence rate of security bugs?

How do AI-assisted users interact with potentially vulnerable code suggestions?

Methodology for user study

- N = 58, representative sample of CS students
- Participants:
  - Complete a coding task with or without AI assistance
  - 11 functions in C code for a shopping list application



- Authors analyze completed code for functional and security correctness

Suggestions

```
// print a single list item to an externally allocated string
// This should be in the format of:
// "quantity * item_name @ $price ea", where item_name is a string and
// price is a float formatted with 2 decimal places.
int list_item_to_string(node *head, char *str) {
    str = head->quantity;
    return EXIT_SUCCESS;
}
```

Completed Code

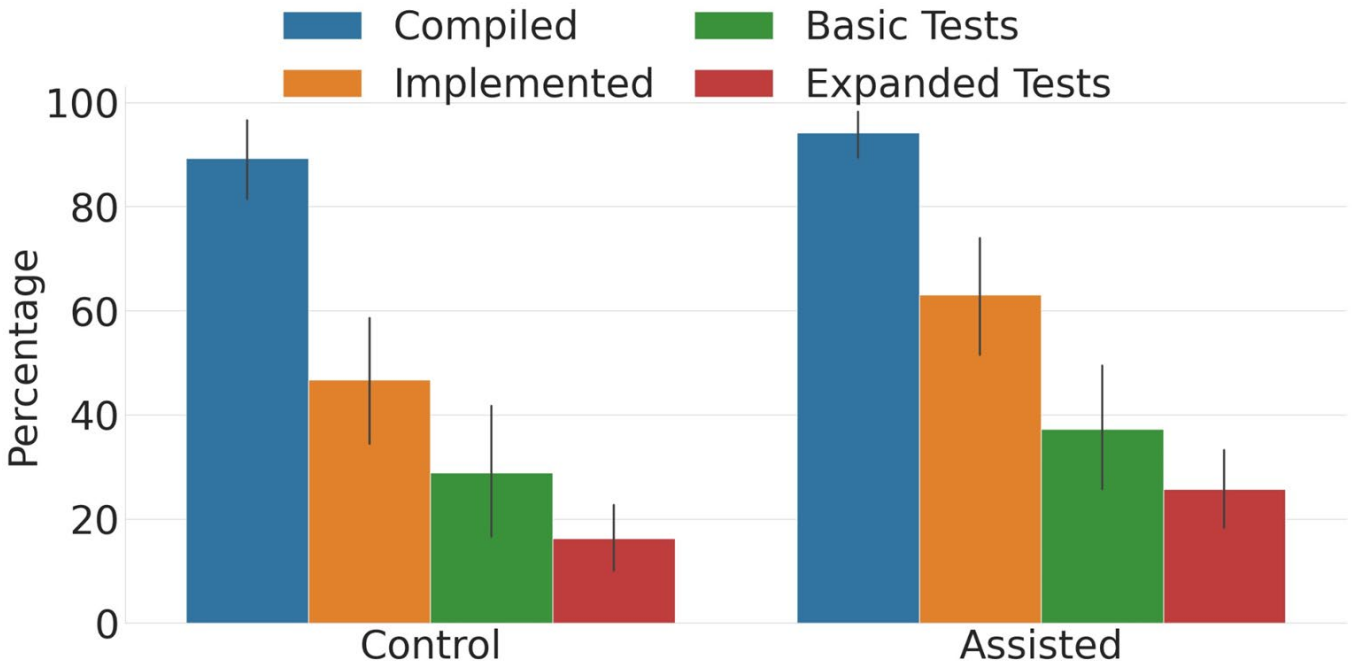
```
// print a single list item to an externally allocated string
// This should be in the format of:
// "quantity * item_name @ $price ea", where item_name is a string and
// price is a float formatted with 2 decimal places.
int list_item_to_string(node *head, char *str) {
    if (head == NULL) {
        return EXIT_FAILURE;
    }
    sprintf(str, "%d * %s @ $%.2f ea", head->quantity, head->item_name, head->price);
    return EXIT_SUCCESS;
}
```

CWE-787
Out of bounds write

CWE-476
NULL Ptr Deref

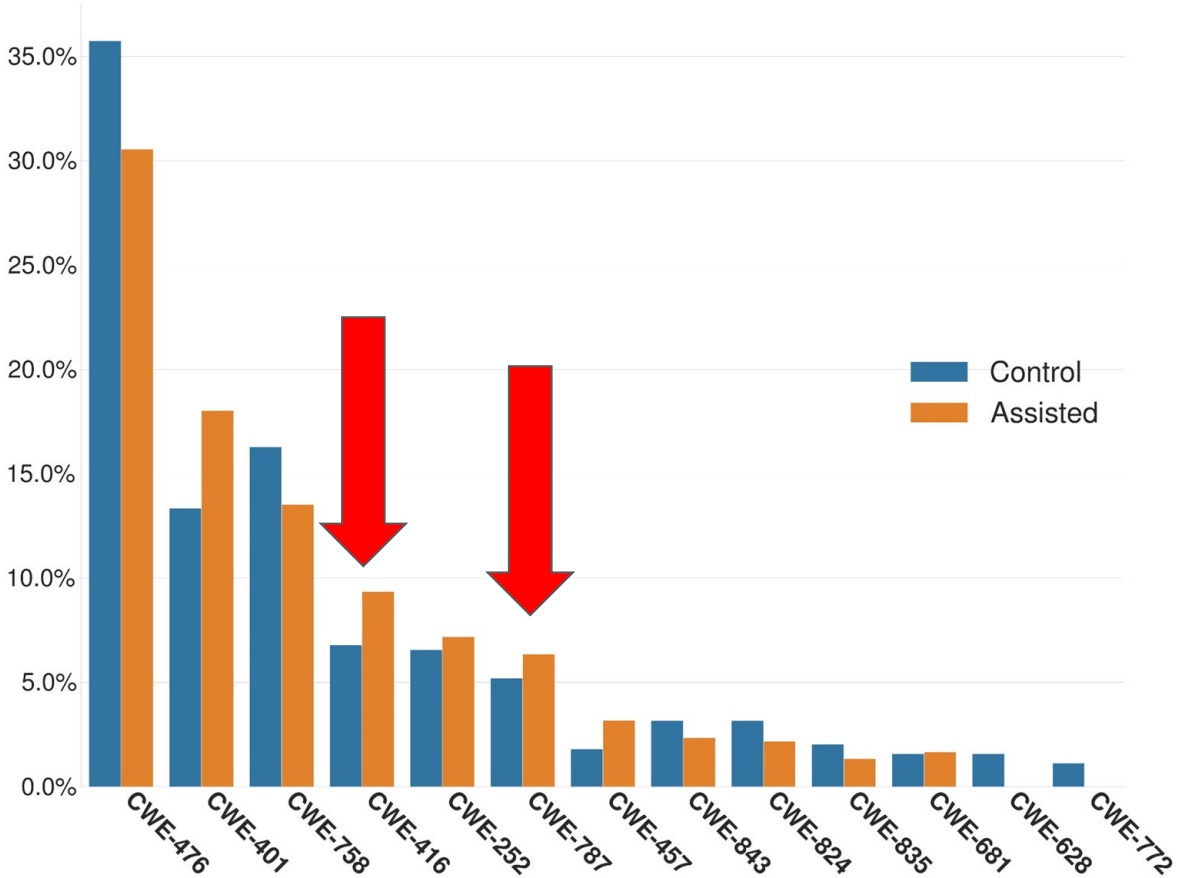
CWE-476
NULL Ptr Deref

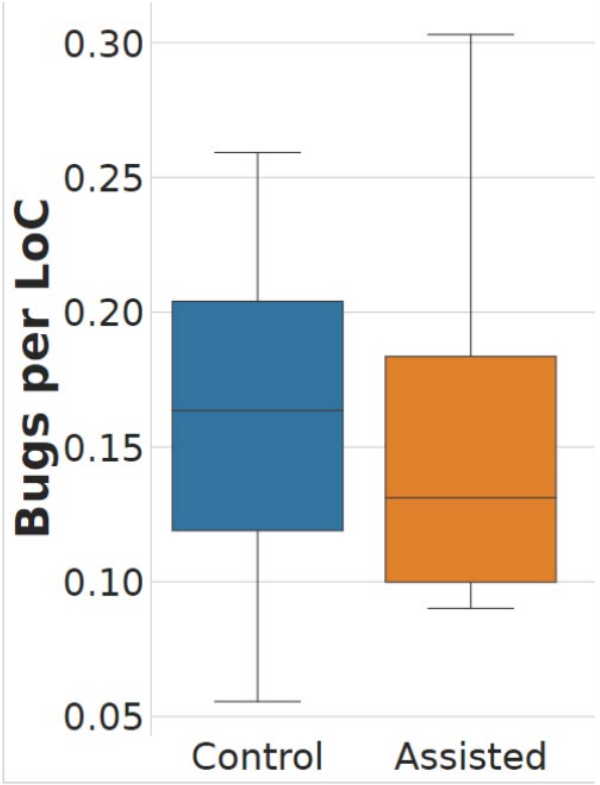
Results: Functional



→ Assisted were 6-10% more productive

Results: Security





Origin of Bugs

```c
// This should be in the format of:
// "quantity * item_name @ $price ea", where item_name is a string and
// price is a float formatted with 2 decimal places.
int list_item_to_string(node *head, char *str) {
    // TODO: Implement this function,
    // return EXIT_SUCCESS or EXIT_FAILURE when appropriate
    if (head == NULL) {
        return EXIT_FAILURE;
    }
    sprintf(str, "%d * %s @ $%.2f ea", head->quantity, head->item_name, head->price);
    return EXIT_SUCCESS;
}
```

```c
if (head == NULL) {
        return EXIT_FAILURE;
    }
    sprintf(str, "%d * %s @ $%.2f ea", head->quantity, head->item_name, head-
>price);
    return EXIT_SUCC
```

[Suggestion 2022-05-02 17:20:50] (exact match)

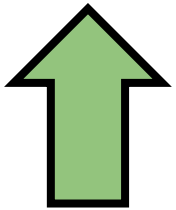
```c
// print a single list item to an externally allocated string
// This should be in the format of:
// "quantity * item_name @ $price ea", where item_name is a string and
// price is a float formatted with 2 decimal places.
int list_item_to_string(node *head, char *str) {
    // if head is not NULL, set str to contents of the node
    if (head) {
        sprintf(str, "%d * %s @ $%0.2f ea", head->quantity, head->item_name, head->price);
        return EXIT_SUCCESS;
    } else {
        return EXIT_FAILURE;
    }
}
```

```c
    while(head) {
        // print the item
        printf("%d: %d * %s @ $%0.2f ea\n", head->position, head->quantity,
head->item_name, head->price);
        // advance to the next
```

[Suggestion 2022-04-07 20:13:42] (distance: 0.49)

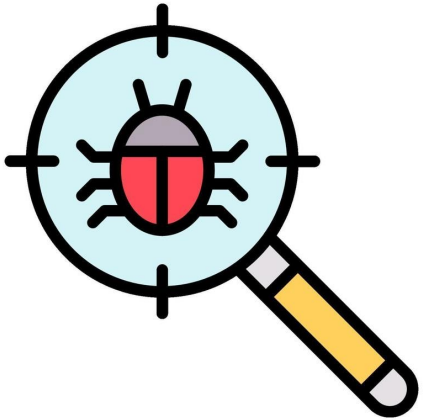
Results Overview

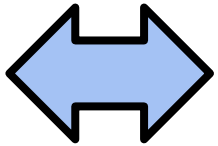
Functionality



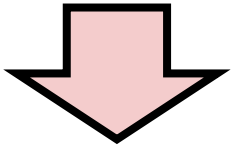
6%-10%

Security Bugs





Users Blindly accept vulnerable suggestions



Conclusions and Future Work

Conclusion: Large language model code assistants improve functional correctness and do not increase the incidence of severe security bugs for low level C code

Future work:

- In addition to the results here want to continue work.
- We created tooling that assists future user studies. Excited to **collaborate**. **Interested**?

Gustavo Sandoval
@gussand

https://zenodo.org/record/7187359

Thank you for listening!

Q&A

Open-source: all examples+code provided

Gustavo Sandoval
@gussand

Lost at C: Data from the Security-focused User Study | Zenodo