Tufts Security & Privacy Lab
SPAM Security & People At Mines

# (Don’t) Trust, but (Don’t) Verify

## How developers evaluate the security of AI-generated code

Download the poster!

**Hamza Khalid** Ronald Thompson Alejandra Sabater Perucy Mussiba Kelsey Fulton Daniel Votipka

**22%** of 400 submissions had no vulnerabilities

**5%** wrote secure code for all four tasks

**21s** spent reviewing each suggestion, on average

**18 / 23** interviewees judged security by clear edge case handling blocks

### 01 / THE QUESTION

## Every mode of AI-assisted development rests on one act: accept, reject, or edit.

Roughly 45% of LLM-generated code still ships vulnerabilities, BUT:
Can the developer catch them?

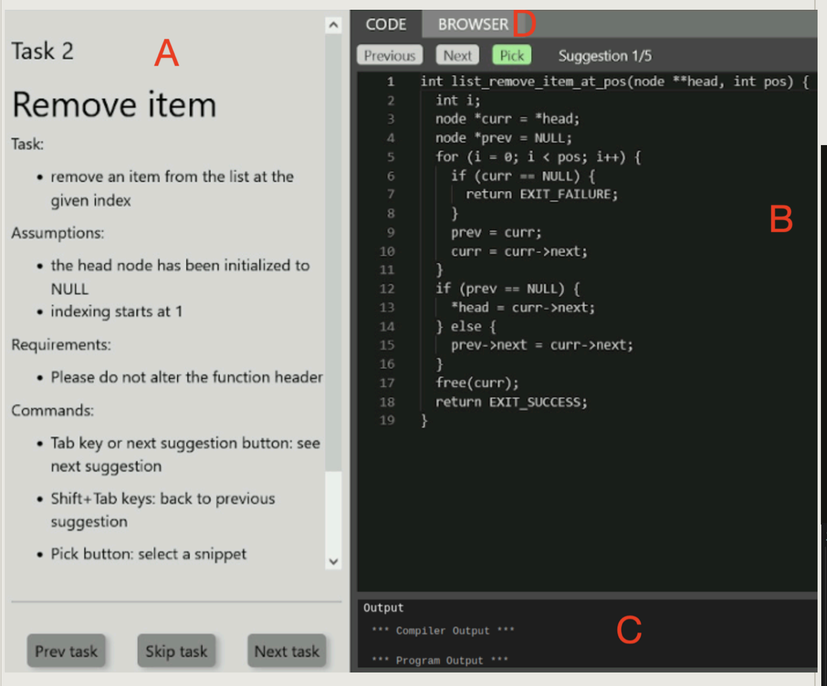
**RQ1** Can they tell secure AI code from insecure?

**RQ2** How do they actually review it?

**RQ3** How does trust shape that review?

### 02 / THE STUDY

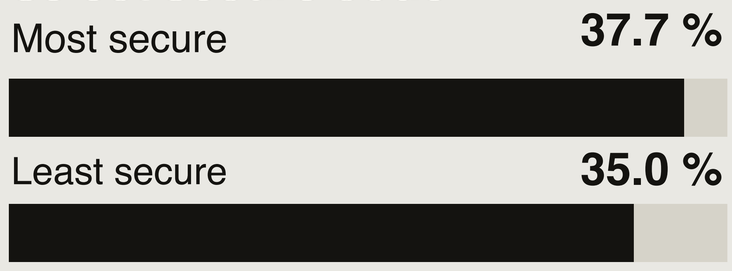
| | |
|---|---|
| **100** | developers, remote, one session |
| **4** | C linked-list tasks: add, remove, update, swap |
| **5** | real AI suggestions per task, varying in security and functionality |
| **7** | years of programming experience on average |
| **2** | years of secure coding experience on average |
| **23** | follow-up interviews, plus 100 surveys |
| **$70** | total compensation including bonus for both secure *and* functional code |




*The NERDS environment where developers took our study.*

### 03 / WHAT THEY PICKED

## Developers could not reliably select secure code




## The pick decided the outcome.

2.0 vulnerabilities from better suggestions, 2.4 from the worst. Did not observe participants picking insecure code and repairing it.

## Editing worked.

29.2 lines changed on average and the more they changed, the fewer flaws survived.

## Experience cut the wrong way.

3+ years of programming predicted *more* vulnerabilities. Security experience predicted nothing.

### 04 / HOW THEY REVIEWED

## A glance, not a review

### 18 of 23 judged security by whether edge cases were visibly handled.

One suggestion with a clearly marked edge-case block still skipped a NULL check, built a circular list and leaked memory. Another, with checks woven through the body, had none.

```c
int swap_item(node **head, int pos1, int pos2) {
    node *curr1, *curr2, *prev1, *prev2;
    int i;
    // check for invalid inputs
    if (head == NULL) return EXIT_FAILURE;

    // check if the list is empty
    if (*head == NULL) return EXIT_FAILURE;

    // check if the position is past the size of the list
    if (pos1 == 0 || pos2 == 0) return EXIT_FAILURE;

    if (pos1 == pos2) return EXIT_SUCCESS;

    curr1 = *head;
    for (i = 1; i < pos1; i++) {
        if (curr1->next == NULL) return EXIT_FAILURE;
        prev1 = curr1;
        curr1 = curr1->next;
```

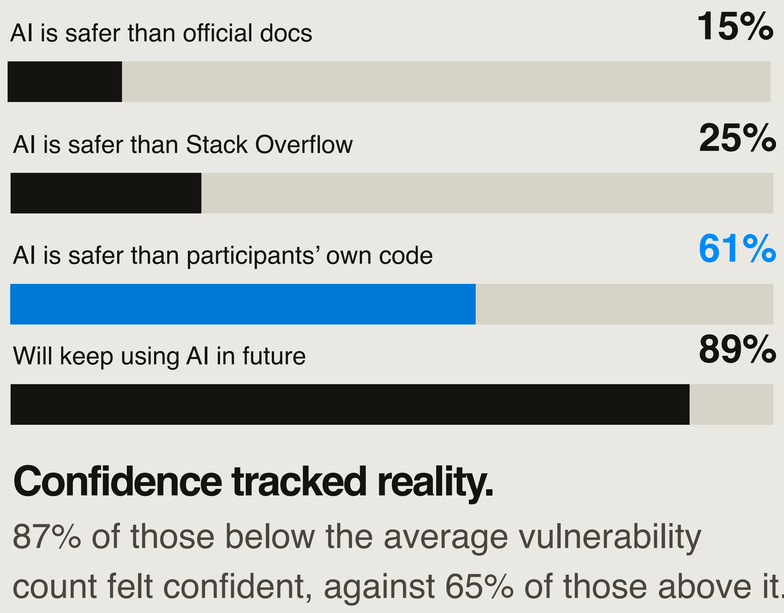
```c
int remove(node **head, int pos) {
  int i;
  node *curr = *head;
  node *prev = NULL;
  for (i = 0; i < pos; i++) {
    if (curr == NULL) {
      return EXIT_FAILURE;
    }
    prev = curr;
    curr = curr->next;
  }
  if (prev == NULL) {
    *head = curr->next;
  } else {
    prev->next = curr->next;
  }
  free(curr);
  return EXIT_SUCCESS;
}
```

**I = 7** put functionality first, security second

**I = 5** judged the task too low-risk to bother

**I = 4** lacked the security knowledge to try

### 05 / TRUST




## Confidence tracked reality.

87% of those below the average vulnerability count felt confident, against 65% of those above it.

### TAKEAWAY

## Manual review is not a control

Even when paid a bonus for security, 78% of submissions contained a security flaw.

### 1. Shrink the unit of review

A whole solution invites skimming.

Suggest a function or a line or mark provenance, so only unvalidated code demands scrutiny.

### 2. Point the AI at the flaws

Let the model hunt and the developer triage OR

Let it build the fuzzing and property-test harnesses nobody sets up by hand.

Email me!