Human vs AI: Insecure Use of Security-Sensitive Functions in AI-generated Code in Comparison to Human-written Code

Authors: JoonKu Lee, Sieun Ju, Seunghoon Woo, Heejo Lee **(jnkuhafnir@korea.ac.kr | heejo@korea.ac.kr)**
Address: Korea University, 145 Anam-ro, Seongbuk-gu, Seoul, South Korea **(https://ccs.korea.ac.kr | +82-2-3290-3638)**

The 41st ACM/SIGAPP Symposium On Applied Computing (SAC 2026)

1. Introduction

1.1 Background

As Large Language Models (LLMs) with coding capabilities are widely adopted for software development, we need assurance that AI-generated code is safe to use. While some works have emphasized the risk of vulnerabilities in such code, most do not explore the underlying causes.

1.2 Aim of Work

We analyze the security of AI-generated code via a comparison with human-written code, focusing on the following aspects:

- How do LLMs utilize **Security-Sensitive Functions** (**SSF**s, potentially insecure functions in the C standard library) in their code?
- How do LLMs create vulnerabilities when generating code?
- To what extent can vulnerabilities caused by SSFs be prevented using different prompt structures?

2. Method

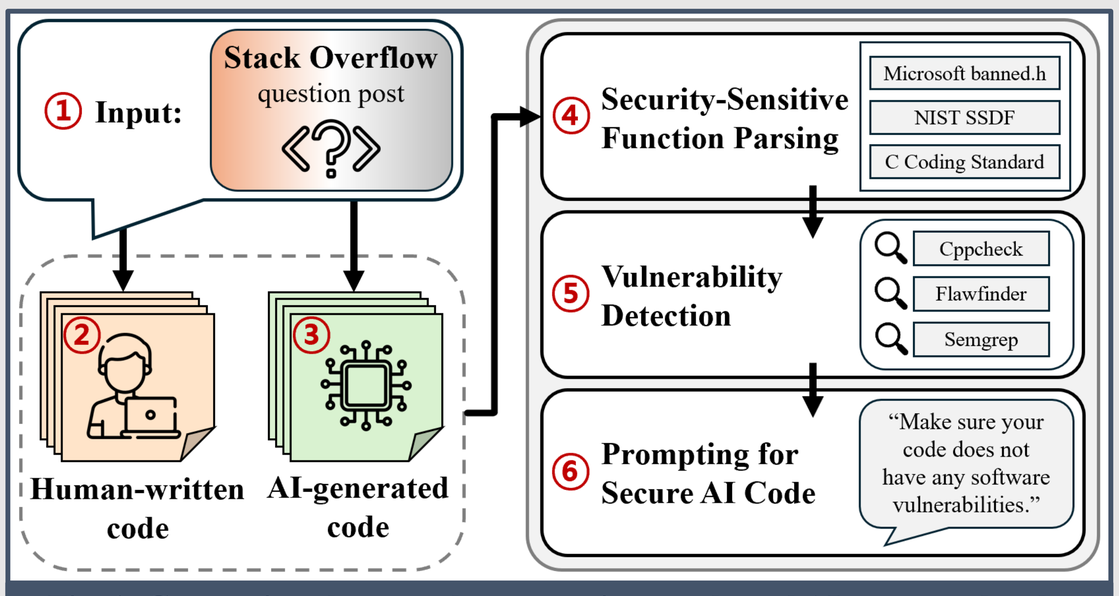

Fig. 1: Comparison between human-written code and AI-generated code.

The comparison process consists of code generation using a common input for both humans and LLMs, and evaluation using static analysis tools.

- Human-written code was obtained from 113 Stack Overflow questions, which were also used to create 339 AI-generated code files.
- GPT-5, Claude Sonnet 4, and Gemini Pro 2.5 were chosen for this study due to their recency, popularity, and code generation capabilities.
- A list of SSFs is defined using resources on secure coding (Fig. 1, ④)
- We selected three SAST tools (Cppcheck, Flawfinder, Semgrep) that can find vulnerabilities without requiring code compilation.

3. Key Findings

3.1 SSF Usage

- More SSFs used in AI-generated code than human-written code.
- Higher SSF usage suggests a <u>wider attack surface of AI-generated code.</u>

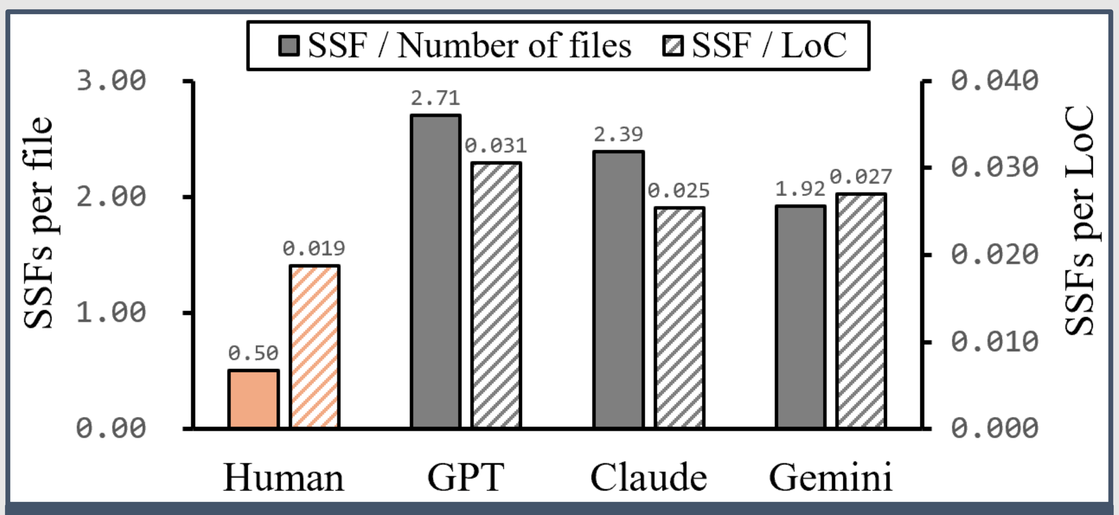

Fig. 2: SSFs used by each code source. (Stack Overflow and three LLMs)

3.2 Vulnerabilities Caused by SSFs

- SSFs in human-written code have a higher vulnerability ratio. (54%)
- For the same code length, AI-generated code is more likely to include vulnerable SSFs than human-written code due to higher SSF usage.
- <u>As a human developer, incorporating and modifying AI-generated code is more dangerous than employing manually written code.</u>

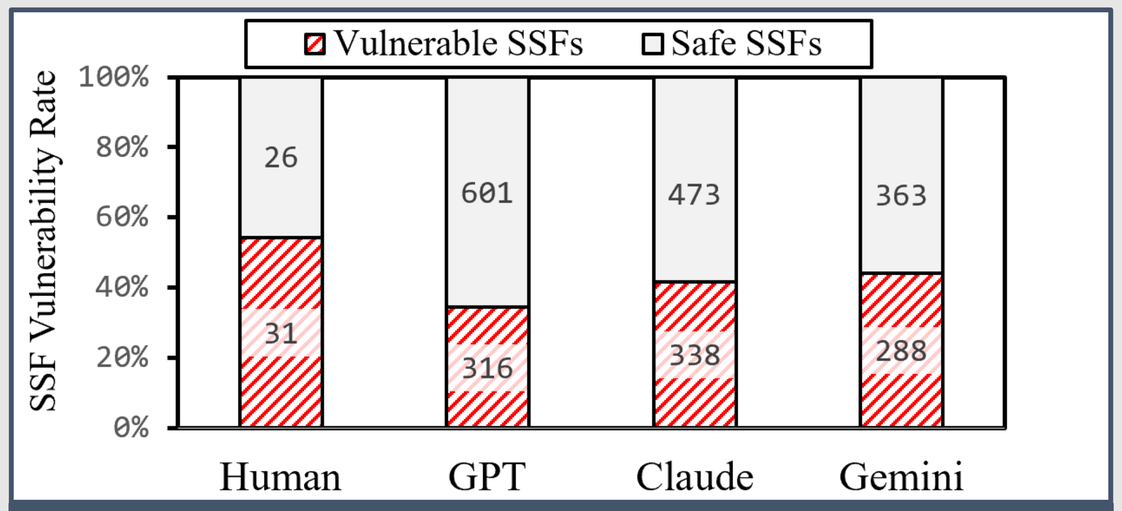

Fig. 3: Number of vulnerable SSFs compared to safe SSFs.

3.3 Code Refinement Using Prompts

- The initial sample of human-written code and AI-generated code is compared with a sample of refined code, created as below:
  - **Manual refinement**: Code from edited Stack Overflow answers, which is patched / improved by the initial answerer.
  - **General refinement**: AI-generated code created by the prompt: *"Make sure your code does not have any software vulnerabilities."*
  - **Specific refinement**: AI-generated code created by prompts specific to a SSF, designed using secure coding resources. E.g.: *"strcpy() does not check for buffer overflows when concatenating to destination. Consider using strcpy_s."*

Original AI-generated code

```
// ...
// Prompts user for file name to use for the copy.
printf("\nWhat would you like to name the duplicate file? (default name: output.txt): ");
fgets(buffer, sizeof(buffer), stdin);
remove_newline(buffer);      // Clean the newline character from input
if(strlen(buffer) > 0) {  // Use user-defined name if provided
  strcpy(destination_file_name, buffer);
}
```

New AI-generated code

```
// ...
// Prompt for destination file
printf("What would you like to name the duplicate file? (default: output.txt): ");
if (fgets(buffer, BUFFER_SIZE, stdin) != NULL) {
  strip_newline(buffer);
  if (buffer[0] != '\0') {
    strncpy_s(destination_file_name, sizeof(destination_file_name), buffer, _TRUNCATE);
  }
}
```

Fig. 4: Comparison of SSF use in initial AI-generated code and refined code obtained by requesting GPT to avoid misuse of *strcpy()* function.

- Some vulnerabilities caused by certain SSFs can be prevented in AI-generated code using prompts requesting the following:
  - Use of a safer alternative function. (e.g., Fig. 4)
  - More thorough error handling. (e.g., boundary checking)
- <u>Other SSFs (such as *memcpy()* or *read()*) cause vulnerabilities regardless of prompt structure</u>.

4. Conclusion

- Our evaluation suggests that for human developers, AI-generated code is more dangerous than human-written code.
  - Higher SSF usage of LLMs indicates a wider attack surface.
  - SSFs in AI-generated code can lead to vulnerabilities when AI-generated code is manually reused or modified.
  - Certain vulnerable SSFs are difficult to prevent using prompts.
- Approaches to mitigate security risks in AI-generated code include training LLMs on code that use SSFs safely, and devising vulnerability detection methods tailored for AI-generated code.