Secode: A Security Testing Framework for AI-Generated Code

Heying Chen*, Chang Zhou*, Yuqian Hong*
*University of Science and Technology of China

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

CONTENTS

Thanks to Tian Xie for his great help in chatgpt script generation

I. INTRODUCTION

Commercial large language models (LLMs) trained on large amounts of source code have been widely promoted as tools to help developers with general coding tasks, such as translating and interpreting code between programming languages, by predicting possible text completion given a few prompts, including comments, function names, and other code elements. Among the many tasks programmers do, we are interested in fixing security vulnerabilities; Developers may often run security tools such as fuzzers or static analyzers in an attempt to understand feedback, locate problems, and modify code to fix bugs.

In this article, we ask the question: Can large language models help us fix security vulnerabilities? Common large language models, such as OpenAI's GPT, are trained on open-source code in a myriad of languages that contain a lot of annotations and features, both buggy and non-buggy. This enhances the LLM's ability to complete the code in different ways, given some context, such as the designer's intent in the code comments.

Recent work has shown that completing code with GitHub Copilot may introduce security vulnerabilities, but Pearce et al. conclude that models can still "increase the productivity of software developers". Since it is possible to guide LLMs by adding hints to the prompts, do we develop large language models for source code security testing feedback?

We wonder if LLMs – despite not being trained on specialized security fixes (and, in fact, on a lot of insecure code) – can still generate effective fixes for vulnerable code. We seek answers to these research questions:

1) Can a paradigm be designed to correct in a multilingual environment and reveal security implications in the code generated by generative LLMs?
2) This solution not only focuses on solving the problem of language-specific generation detection, but also focuses on the research and practical application of generalized generative code security.

To answer these questions, we evaluate recent ChatGPT synthetic or hand-crafted buggy scenarios. Our contributions are as follows:

1) Secode, a standardized security detection scheme for intelligently generated code, was proposed. We have designed correctness and security solutions that can be used in multiple languages.

2) For the security detection of a certain language, we propose how to intelligently generate test code based on common vulnerability datasets. Detect if generative models create specific types of vulnerabilities without misbootstrapping as much as possible.
3) A statistical model for normalized efficiency evaluation was proposed.

II. Ease of Use

A. Maintaining the Integrity of the Specifications

The IEEEtran class file is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

III. Prepare Your Paper Before Styling

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections III-A–III-E below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not number text heads—LaTeX will do that for you.

A. Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

B. Units

- Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as "3.5-inch disk drive".
- Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
- Do not mix complete spellings and abbreviations of units: "Wb/m^2" or "webers per square meter", not "webers/m^2". Spell out units when they appear in text: ". . . a few henries", not ". . . a few H".
- Use a zero before decimal points: "0.25", not ".25". Use "cm^3", not "cc".)

C. Specific Vulnerable Program Repair

```
char* multiply_doubles_str(double d1, double d2) {
    double d3 = d1 * d2;

    // Convert the double to string
    char* d3_str =
    (char*)malloc(sizeof(char) * (DBL_DIG + 1));
    snprintf(d3_str, DBL_DIG + 1, "%lf", d3);

    // Return the string
    return d3_str;
}

int main(int argc, char *argv[]) {
    // Check if the correct number of
    // command-line arguments is provided
    if (argc != 2) {
        fprintf(stderr,
            "Usage: %s <buffer_size>\n", argv[0]);
        return 1;
    }

    // Get buffer size from argv
    int buffer_size = atoi(argv[1]);

    // Allocate buffer
    char *buffer = (char *)malloc(buffer_size);

    // Check if memory allocation is successful
    if (buffer == NULL) {
        fprintf(stderr, "Memory allocation failed.\
        return 1;
    }

    // Read from stdin
    printf("Enter data: ");
    fgets(buffer, buffer_size, stdin);

    // Print buffer
    printf("Buffer content: %s\n", buffer);

    // Free buffer
    free(buffer);

    return 0;
}

from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return '<p>Hello World!</p>'
```

```
# function which says hello to the user
@app.route('/user/<name>')
def user(name):
    return '<h1>Hello, %s!</h1>' % name
    # return '<p>Hello, %s!</p>' % name

# run server
if __name__ == '__main__':
    app.run(debug=True)
```

The three codes shown above are the ones we designed for a specific vulnerability, as well as the fixes that the large language model generates when faced with the vulnerable code.

D. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

E. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is [7].

F. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

G. Releted Works

Pearce et al.(Pearce et al., 2022 [1]) conducted a systematic code generation of Copilot. The approach they used was to generate code based on the pre-existing vulnerable code scenarios, such as vulnerabilities in the CWEs. By giving some incomplete code blocks as prompts for Copilot to generate code suggestions, they then analyzed the suggested codes for security using a combination of CodeQL and manual inspection. They found that Copilot offered suggestions of vulnerable code in about 40%

H. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 1", even at the beginning of a sentence.

TABLE I
TABLE TYPE STYLES

Table Head	Table Column Head		
	Table column subhead	*Subhead*	*Subhead*
copy	More table copy[a]		

[a]Sample of a Table footnote.

Fig. 1. Example of a figure caption.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization {A[m(1)]}", not just "A/m". Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

REFERENCES

[1] Pearce, H., B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri 2022, May.*Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions.*In 2022 *IEEE Symposium on Security and Privacy* (SP), pp.754–768. ISSN: 2375-1207.*IEEE Symposium on Security and Privacy*
https://doi.org/10.48550/arXiv.2108.09293

[2] Majdinasab, V., M. J. Bishop, S. Rasheed, A. Moradidakhel, A. Tahir, and F. Khomh, 2023, Nov.*Assessing the Security of GitHub Copilot Generated Code – A Targeted Replication Study.*
https://doi.org/10.48550/arXiv.2311.11177

[3] Asare, O., M. Nagappan, and N. Asokan, 2024, Jan. *Is GitHub's Copilot as Bad as Humans at Introducing Vulnerabilities in Code?* in *Empirical Software Engineering.*
https://doi.org/10.48550/arXiv.2204.04741

[4] Pearce, H., B. Tan, B. Ahmad, R. Karri, and B. Dolan-Gavitt, 2022, Aug.*Examining Zero-Shot Vulnerability Repair with Large Language Models* in 2023 *IEEE Symposium on Security and Privacy (SP).*
https://doi.org/10.48550/arXiv.2112.02125

[5] Prenner, J. A., H. Babii, and R. Robbes, 2022, Oct.*Can OpenAI's codex fix bugs?: an evaluation on QuixBugs* in *2022 IEEE/ACM International Workshop on Automated Program Repair (APR)*
https://doi.org/10.1145/3524459.3527351

[6] Xu, F.F., U. Alon, G. Neubig, and V.J. Hellendoorn 2022, June. *A systematicevaluation of large language models of code.* In Proceedings of *the 6th ACMSIGPLAN International Symposium on Machine Programming*, San Diego CA USA, pp. 1–10. ACM.
https://doi.org/10.48550/arXiv.2202.13169

[7] Pearce, H., B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri, 2022, Jun.*An Empirical Evaluation of GitHub Copilot's Code Suggestions* in *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*
https://doi.org/10.1145/3524842.3528470