Step-by-Step Vulnerability Detection using Large Language Models

Saad Ullah 1  Ayse Coskun 1  Alessandro Morari 2  Saurabh Pujar 2  Gianluca Stringhini 1

1Boston University  IBM Research

Motivation

- Vulnerability detection is a very critical task for systems security.
- Current analysis techniques suffer from the trade-off between coverage and accuracy.
- ML-based analysis tools are non-robust, black-box and unreliable to use in real-world [1].
- LLMs demonstrate revolutionary capabilities for programming language-related tasks but they are also studied in a black-box fashion for both vulnerability detection and its repair.
- Security experts follow a step-by-step approach for vulnerability detection. Can using the same approach help LLMs performing better at the vulnerability detection task?

Objective

Design a framework to emulate step-by-step reasoning process of a human security expert using LLMs, to efficiently detect vulnerabilities in source code.

Methodology

Our approach uses few-shot in-context learning to guide LLMs to follow a step-by-step human-like reasoning model for vulnerability detection.
- We make sure that the model first generates chain-of-thought reasoning [5] and then makes a decision based on that reasoning (as shown in Figure 1 and 3b).

Figure 1. Overview of our few-shot in-context learning approach for vulnerability detection using LLMs.

Evaluation

- Figure 3 shows that step-by-step reasoning guides the LLM to detect the (CWE-787) vulnerability.
- To systematically evaluate this approach, we create our own diverse synthetic dataset based on a subset of the MITRE 2022 top 25 most dangerous vulnerabilities.
- For each vulnerability we create vulnerable examples and their patches with varying levels of complexity.
- We use the "gpt-3.5-turbo-16k" chat API to compare our approach with SoTA tools (Table 1).

Table 1. Evaluation of different vulnerability analysis techniques on our dataset.

<table>
<thead>
<tr>
<th>Tool/Model</th>
<th>Description</th>
<th>Size</th>
<th>F1</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>cppcheck, infer, flawfinder</td>
<td>Combination of SoTA static analysis (SA) tools for C/C++</td>
<td>-</td>
<td>0</td>
<td>0.53</td>
</tr>
<tr>
<td>UniXcoder</td>
<td>RoBERTa-based model fine-tuned for defect detection in C/C++</td>
<td>126M</td>
<td>0</td>
<td>0.25</td>
</tr>
<tr>
<td>CodeS5+</td>
<td>LLM specifically pre-trained for programming languages-related tasks, including C/C++</td>
<td>16B</td>
<td>0</td>
<td>0.54</td>
</tr>
<tr>
<td>GPT-3.5</td>
<td>LLM with no reasoning</td>
<td>175B</td>
<td>0</td>
<td>0.72</td>
</tr>
<tr>
<td>Our approach with GPT-3.5</td>
<td>GPT-3.5 with step-by-step reasoning</td>
<td>175B</td>
<td>0</td>
<td>0.70</td>
</tr>
</tbody>
</table>

Tookaways

- Following a human-like step-by-step reasoning approach helps LLMs to efficiently analyze code and detect vulnerabilities.
- Our approach provides an explanation for the detected vulnerabilities, which helps users to better contextualize them and to find their root cause.
- Systematic evaluation of this approach on real-world datasets is still required to determine its reliability in real-world use cases.

References