Project Title

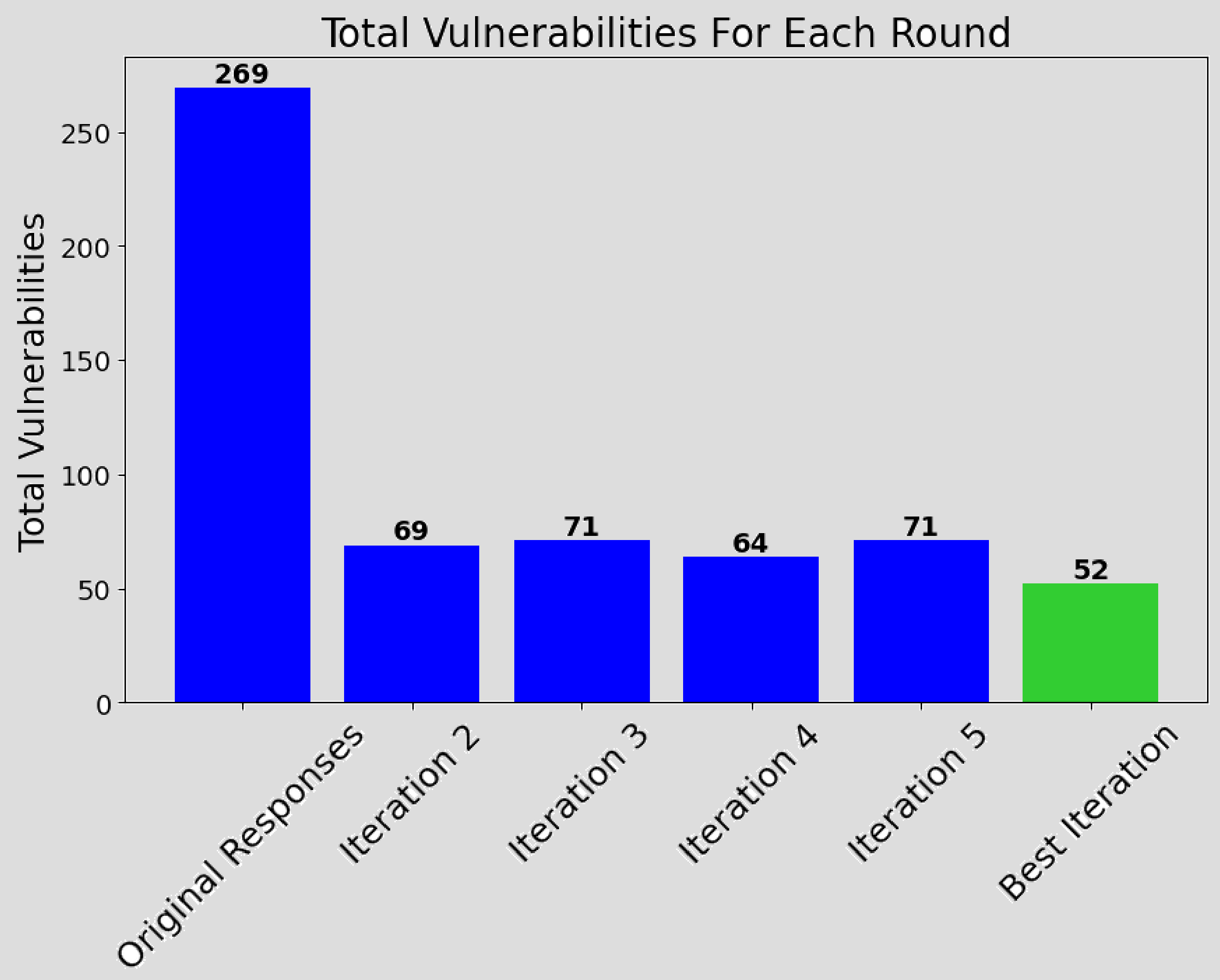
Jake Milne
Supervised By Dr Kyle Martin

Aims & Objectives

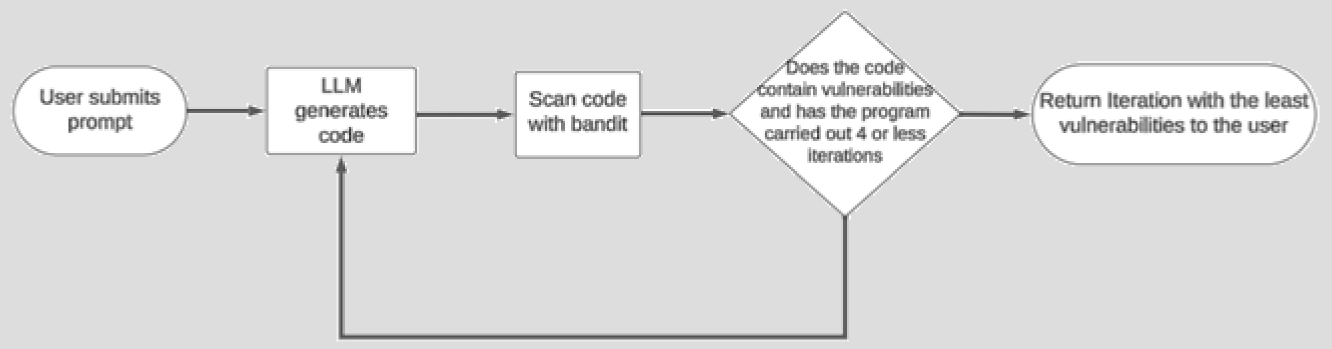
Since the release of ChatGPT in 2022, the usage of Large Language Models (LLMs) in code creation has become inescapable. Stack Overflow's 2024 developer survey found that 75.6% of all respondents either already used or planned to start using Artificial Intelligence (AI) tools in the development process. Despite this, LLM generated code has earned a reputation for containing bugs and vulnerabilities.

The goal of this project is to develop an automated pipeline which identifies and removes vulnerabilities present in Large Language Model (LLM) generated code. This will be achieved by creating an iterative system which identifies and locates any vulnerabilities and attempts to regenerate the code with the vulnerabilities removed.

This project will be considered a success if there is a noticeable decrease in vulnerabilities between the first and final rounds of code generation



Methods


User submits prompt
LLM generates code
Scan code with bandit
Does the code contain vulnerabilities and has the program carried out 4 or less iterations
Return Iteration with the least vulnerabilities to the user

This project will be implemented as a web-app to handle user interaction, with a java backend which was used to communicate with the LLMs and a docker container with python3 and bandit installed.

1. Generate the initial code using only the user's prompt. This stage just involves getting the users input from the frontend and supplying it to the LLM. The user is also able to supply a test case
2. Scan the code using bandit. The response from the LLM is then parsed to retrieve the code. This is then saved inside a docker container, bandit is then used to identify vulnerabilities in the code. If the user has also specified a unit test, the code is also run.
3. Regenerate the code if vulnerabilities are found and/or if a unit test fails, using the vulnerability/failure as a reference.
4. The iteration with the least vulnerabilities is then given to the user.

This process will finish when a response either contains no issues or if vulnerabilities still exist after 5 iterations

OpenAI's GPT-4o-Mini is used for the first two rounds (initial generation and the first re-generation used the results from bandit), and GPT-4o is then used for the remaining rounds.

Results

Testing was carried out using the LLMSecEval Prompts Dataset, Each prompt in the dataset was tested three times. The first iterations (the original responses) contained a total of 269 vulnerabilities, this was reduced to 69 after 1 iteration, and then to 52 after all iterations had completed, resulting in an 80.7% reduction. Of the remaining 52 vulnerabilities; CWEs 78 and 22 appeared 46 and 6 times respectively, making up all the remaining vulnerabilities. These CWE 78 relates to OS command injection, and CWE 22 relates to Path traversal, Due to the nature of the prompts used to produce them, these vulnerabilities may be unavoidable without breaking the functionality specified in the prompt.

CWE No.	First Iteration Count	Best Iteration Count	Difference (%)
94	116	0	100
78	64	46	28.15
330	39	0	100
259	35	0	100
22	8	6	25
377	6	0	100
327	1	0	100

Acknowledgements

Vitam voloria con et, cum laboreium, nam cum, quat paris as core volorest, sust exceatur antio ea voluptas eumquatendit quas doluptatet aut landignam que lantur sania dem escidit min reruptiaero velis maximus aut que pe aliquas sequidisit, omnient hit lam eicias modio con nisin pediam, ent, cust accaborit odio corehen dendae nonet eliquam, aperorepe autatem voles int odic tem nis dolorrum que cores doluptaturi utemolorum imusam, idendaecabo.

RGU ABERDEEN SCHOOL OF COMPUTING, ENGINEERING & TECHNOLOGY

BSc (Hons) Computer Science