ChatGPT as a Code-Security Analysis Engine:
A Comparative Evaluation

CyberVSR 2023
Katherine Carlile

Use cases of ChatGPT

OpenAI’s large language model

Designed to excel in conversation

Excels in conversation scenarios

Prefers long, exhaustive answers over concise replies

ChatGPT as a Cybersecurity tool

In recent years, researchers have explored the application of ChatGPT as a cybersecurity tool to some success

Technical use cases include: code creation, code documentation, bug detection, refactoring, etc.

PentestGPT

- ChatGPT-powered penetration testing tool, capable of solving beginner to intermediate Capture The Flag challenges

Static Code Scanner

- GPT-3, text-davinci-003, identified 213 security vulnerabilities in a vulnerable source code repository

Motive

- ChatGPT has seen a tremendous increase in popularity, and a rapidly growing breadth of use-cases
- Application into computer science and cybersecurity has been promising, but not without limitations
- How would ChatGPT compare as a vulnerability assessment tool against a commercially available tool purpose-built for the application?

CodeQL

GitHub’s machine learning-powered semantic code analysis engine

- Used to detect vulnerabilities in a codebase
- CodeQL queries the codebase for vulnerabilities, from a standard or customized query list

Experiment Setup

Dataset:

- CrossVul: A Cross-Language Vulnerability Dataset
  - Contains code with known MITRE Common Weakness Enumerations (CWEs) and patched-file counterparts

Cleaning Procedure:

- Only Javascript files
  - JS is the language behind common exploit attempts
- MITRE's 2023 Most Dangerous Software Weaknesses

Dataset Result:

- 16 CWE folders, containing ~10 vulnerable code samples per folder

Dataset passed through CodeQL and ChatGPT (gpt-3.5-turbo) tools

ChatGPT Results

Mixed Success

- ChatGPT was able to evaluate all CrossVul_JS files and report any found weaknesses, but not without issue:
  - Would not report weaknesses on files known to contain weaknesses
  - Would report weaknesses other than the weakness known to be contained in the file

Challenges:

- Rate and Token Limits
  - Solved by removing past model context, implementing exponential backoff, and applying a max token limit
- Prompt Engineering
  - Model would deviate from prompt directions, which brings into question the reliability of results

CodeQL Results

Mixed Success

- Reported the correct weakness in a file corresponding to that CWE
- Failed to find all weaknesses in the repository files

Challenges

- Identification Failure:
  - Standard/Non-specific usage: Repository was scanned using GitHub's built-in CodeQL tool rather than an IDE
  - Using a standard query, instead of a Javascript CWE-specific query list

Comparison

CodeQL and ChatGPT were able to evaluate files for CWEs, with mixed success

- CodeQL accurately identified CWEs in the codebase, but with many false negatives
- ChatGPT identified CWEs in the codebase, in addition to false negatives, false positives, and deviations from the prompt

Conclusion

ChatGPT is a large language model best suited to conversational use-cases, and struggles with factual evaluations

- Prone to deviations from prompts and hallucinations
- Requires strict management to perform desired task as intended

CodeQL is purpose-built for static code analysis, and excels at accurately identifying known vulnerabilities

- May be less adaptable to new vulnerabilities - yet unidentified but similar to known vulnerable-code patterns

Next Steps:

- Improve ChatGPT prompt engineering and ensure necessary context is maintained through chat completion
- Implement Javascript-specific CWE query list for CodeQL, using Visual Studio Code