School of Computing, Engineering, and the Built Environment
Edinburgh Napier University

PHD STUDENT PROJECT

Application instructions:
Detailed instructions are available at :
https://www.napier.ac.uk/research-and-innovation/doctoral-college/how-to-apply

Prospective candidates are encouraged to contact the Director of Studies (see details below) to discuss the project and their suitability for it.

Project details

Supervisory Team:

- DIRECTOR OF STUDY: Dr Kehinde Babaagba (Email: k.babaagba@napier.ac.uk)
- 2ND SUPERVISOR: tbc

Subject Group: Computer Science

Research Areas: Computer Science

Project Title: Large Language Models for Vulnerability Detection

Project description:

As software systems become increasingly complex, the demand for automated vulnerability detection has never been higher. Software vulnerabilities, such as buffer overflows, SQL injections, and cross-site scripting, pose significant risks to security, privacy, and the overall integrity of digital infrastructure. Traditional approaches to vulnerability detection often rely on static and dynamic analysis tools that require significant manual effort, domain expertise, and are frequently unable to generalize across diverse codebases. The recent advancements in machine learning, particularly in the field of Natural Language Processing (NLP), offer a promising avenue to address these challenges. Large Language Models (LLMs), such as GPT-4 and its successors, have demonstrated impressive capabilities in understanding and generating human-like text, making them well-suited to analyse and understand programming languages, which are structured similarly to natural languages.

The primary objective of this Ph.D. project is to investigate the application of LLMs for detecting vulnerabilities in software code. The research will focus on exploring how these models can be trained or fine-tuned to recognize potential security flaws in a wide range of programming languages, understand the context in which vulnerabilities arise, and suggest possible mitigations or corrections. The key goals include:

1. Model Development and Fine-tuning: Develop and fine-tune LLMs specifically for the task of vulnerability detection. This will involve curating large datasets of code containing both vulnerable and non-vulnerable examples, as well as leveraging transfer learning techniques to adapt general-purpose LLMs to this domain-specific task.

2. Contextual Understanding: Enhance the LLMs' ability to understand the broader context of code, such as the dependencies between different components, libraries, and APIs, which are crucial for accurately identifying vulnerabilities that may not be evident from a local analysis alone.

3. Multi-language Support: Extend the capability of LLMs to detect vulnerabilities across different programming languages. This will require the development of multi-lingual models or the creation of language-agnostic features that can generalize across various programming languages and paradigms.

4. Evaluation and Benchmarking: Rigorously evaluate the performance of the developed models against existing state-of-the-art vulnerability detection tools. This will involve the creation of comprehensive benchmarks that include a diverse set of real-world and synthetic code samples, covering a wide range of vulnerability types.

5. Explainability and Interpretability: Develop methods to make the predictions of LLMs more interpretable to developers and security analysts. This includes generating human-readable explanations for why certain code segments are flagged as vulnerable and how they can be fixed, thus bridging the gap between automated detection and human expertise.

6. Integration with Development Workflows: Explore how the developed models can be integrated into existing software development and continuous integration/continuous deployment (CI/CD) pipelines. This would enable real-time vulnerability detection as part of the software development lifecycle, providing developers with immediate feedback on potential security issues.

   Methodology
   The research will adopt a multi-faceted methodology combining theoretical model development with empirical experimentation. Key components of the methodology include:

   1. Data Collection and Preprocessing: Curate a large-scale dataset of code snippets and full programs annotated with known vulnerabilities. This dataset will serve as the foundation for training and evaluating LLMs. Data preprocessing will involve tokenization, normalization, and possibly the anonymization of sensitive information.

   2. Model Training and Fine-tuning: Utilize transfer learning to fine-tune pre-trained LLMs on the vulnerability detection task. This may involve

experimenting with different architectures, such as transformer-based models, and exploring the impact of varying the size and structure of training datasets.

3. Experimental Evaluation: Conduct extensive experiments to evaluate the performance of the models on both standard benchmarks and real-world codebases. Metrics such as precision, recall, F1 score, and the number of false positives/negatives will be used to assess the effectiveness of the models.

4. User Studies and Feedback: Conduct user studies to gather feedback from developers and security experts on the usability and effectiveness of the LLM-generated vulnerability reports and recommendations. This feedback will inform further refinements to the models.

Candidate characteristics

Education:

The ideal candidate should have a first degree with at least a 2:1 classification in one of the following subjects: Computer Science, Cybersecurity, Artificial Intelligence/Machine Learning, Software Engineering, Data Science or similar subjects.

Subject knowledge:

The ideal candidate should have a strong foundation in Computer Science and Software Engineering, with a deep understanding of Programming Languages and Data Structures and Algorithms. They should possess solid knowledge of Machine Learning, particularly in Natural Language Processing (NLP), and be well-versed in Cybersecurity principles, including common vulnerabilities and secure coding practices. Familiarity with the Software Development Lifecycle is essential, as is experience with handling and analysing large datasets.

Essential attributes:

- Strong Programming Skills
- Understanding of Machine Learning
- Knowledge of Cybersecurity
- Analytical and Problem-Solving Skills
- Research Aptitude
- Data Management Skills
- Attention to Detail
- Good Communication Skills
- Self-Motivation and Initiative
- Collaboration Skills
- Adaptability and Willingness to Learn
- Critical Thinking