大規模言語モデルによるソフトウェア脆弱性の検出

高知大学
理工学部情報科学科
横川武典

1. 研究背景

言語モデルがコード生成やデバッグ等に活用

→ 生成されるコードの**安全は担保されていない**

図1. コーディングに使われる言語モデルが含まれるツールの例

2. 研究目的

言語モデルで脆弱性を**自動で検知**
以下2つの問題に対応可能

1. 言語モデルが生成したコードは**安全と限らない**
   → コードに自動で脆弱性の検知を行い安全性を確保
2. 脆弱性を発見には**テスト**や**コードレビュー**が必要
   → 手間の削減が可能

3. 関連研究

LLMs in Software Security: A Survey of Vulnerability Detection Techniques and Insights [Ze Sheng+2025]

- **特定のコードの断片**からは、脆弱性を検出できる言語モデルが存在
- **リポジトリ**からなるアプリケーションでは、限定的な脆弱性の検出のみ可能
- メモリ関連の脆弱性は検出精度が高い
- C/C++に関する研究が多い

4. 研究手法

脆弱性のデータベースを取得
データベースを参照/学習に使い脆弱性を発見可能に

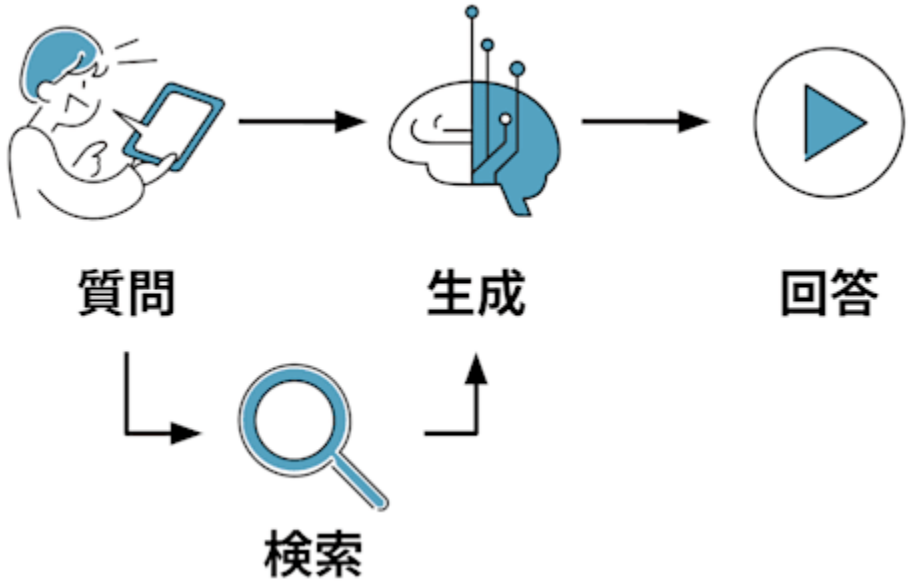
1. 脆弱性のデータベースを取得/作成
2. データベースを元に**RAG**の作成/**FineTuning**
3. ソースコードから脆弱性を探す

RAG(Retrieval-Augmented Generation)とは

RAGは**事前学習していない外部知識を検索しその情報に基づいて文章を生成する手法**

以下の3段階で機能する

1. クエリと**類似する情報**をデータベースから検索
2. **得られた情報**をプロンプトに追加
3. LLMにプロンプトを入力して推論



FineTuningとは

既存の学習済みモデルに、**追加の学習を行い特定のタスク用に調整する**手法

メリットは以下の3つ

1. 一から学習しないため**コストが安い**
2. 最新の情報への対応
3. 特定の用途に**特化したモデル**を作成可

5. 実験計画

1. 脆弱性のデータベースを取得/作成

以下の脆弱性の情報をまとめたデータベースを作成する

1. 要約
2. 内容
3. 影響を受けるソフトウェア

2. データベースを元にRAGの作成/FineTuning

- RAGを作成する場合
  自然言語で書かれた脆弱性のデータベースから**ベクトル**に変換
  この処理はPythonの"LangChain"等のライブラリを使用

- FineTuningの場合
  オープンソースで公開されている言語モデルに脆弱性のデータベースで**追加学習**
  この処理はPythonの"unsloth", "trl"等のライブラリを使用

3. ソースコードから脆弱性を探す

作成したRAGを利用する言語モデルまたはFineTuningした言語モデルで脆弱性を探す

脆弱性を探す対象は以下の2つ

- 既存のソースコード
- 言語モデルの生成したソースコード

6. 現状の進捗

6-1. データの取得

以下の情報を元にデータベースを構築

- 米国政府の非営利団体MITRE Corporationが管理する脆弱性及び識別番号**CVE**
- 日本で利用されるソフトウェアの脆弱性関連の情報をまとめた**JVN**

WordPressに関する脆弱性を取得しjsonで保管

6-1. データの取得

以下にjson形式で集めたデータの一例を示す

このデータベースを活用し、**脆弱性を発見すること**が目標

```
"JVNDB-2025-009951": {
  "title": "AntoineH の WordPress 用 Football Pool における…",
  "description": "The Football Pool plugin for WordPress is …",
  "technologies": "AntoineH Football Pool 2.12.5 未満"
  },
```

図2. 脆弱性データベースの一例

6-2. 脆弱性の発見

調整していない言語モデルに脆弱性を探させてみる

1. オンラインのモデルとして"chatGPT-4o mini"を利用
   "chatGPT-4o mini"で生成したphpのコード全文から脆弱性を探させる
   → 簡単なコードの脆弱性は**検出/修正可**
2. ローカル用モデルとして"gpt-oss-20b"を利用
   "chatGPT-4o mini"で生成したphpのコード全文から脆弱性を探させる
   → 結果は**出力が不安定に**
      → 入力長が長いことが原因

7. 今後の課題

1. 入力長が長くても安定した出力を得られる言語モデルを探す
2. 脆弱性情報を元に言語モデルを**FineTuning**
   または**RAG**等を作成し言語モデルが情報を参照可能に
3. どの程度の脆弱性を発見できるか調査

参考文献

- [Ze Sheng+2025] LLMs in Software Security: A Survey of Vulnerability Detection Techniques and Insights
  https://arxiv.org/html/2502.07049v2
- JVN iPedia - 脆弱性対策情報データベース
  https://jvndb.jvn.jp/
- CVE: Common Vulnerabilities and Exposures
  https://www.cve.org/CVERecord

Appendix A. WEB言語の脆弱性の傾向

2022年に脆弱性が悪用された技術としてWordPressがトップ
全体の1/4を占めた