G-24

LLM を用いた Peach Fuzzer のファジング設定支援手法の提案
Proposal for an LLM-Assisted Fuzzing Configuration Method for Peach Fuzzer

丸岡 哲也†　　井口 信和‡§
Tetsuya Maruoka　Nobukazu Iguchi

1. 序論

令和 6 年度の総務省の情報通信白書によると，サイバー攻撃関連通信数は年々増加傾向にあり，サイバーセキュリティ上の脅威は増大している．サイバー攻撃の増加に伴い，システムに存在する脆弱性を発見し，対処するためのセキュリティテストの重要性が高まっている．また，生成 AI の一つである大規模言語モデル(以下，LLM)の発展が著しく，注目を集めている[1]．LLM は膨大なテキストデータによってトレーニングされたモデルであり，質問への応答，文章の要約や翻訳，ソースコードやドキュメントの生成などの能力を持つ．

セキュリティテストには，ソフトウェアのソースコードやバイナリを解析する静的テストや，動作しているシステムを実際に検証・評価するペネトレーションテストやファジングテストなどの動的テストがある．ファジングテストはファズと呼ばれる不正，無効，ランダムなデータをテスト対象となるシステムに与えることで，その応答や挙動から脆弱性を発見する手法である．

ファジングツールの一つに，Peach Fuzzer 社が開発した「Peach」がある[2]．Peach は複数の OS(Windows，Linux，Mac OS X)上で動作し，図 1 に示すような Pit と呼ばれる XML 形式の設定ファイルを記述することで，TCP/IP などで通信するソフトウェア，Web アプリケーションなど，様々なシステムに対してファジングを実行することができる[3]．また，Community edition は MIT ライセンスでオープンソースソフトウェアとして公開されていること，Pit ファイルのサンプル入手が容易であること，seed 値を指定できることから異なる seed 値を指定しテストの無作為性を担保可能であること，逆に seed 値を固定することで例外の発生を容易に再現できることなど，様々な利点がある．しかし Peach には，ファジング実行前の Pit ファイル記述において，プロトコルの詳細なデータ構造の理解とファジングの知識が必要であり，その記述量も非常に多いため，ファジング作業に多大な負担が生じるという問題点がある[4]．

そこで本研究では，Peach を使用するファジング作業者の負担軽減を目的に，LLM を用いて，従来の Pit の記述に必要なプロトコルの詳細なデータ構造などに比べて作業者が理解しやすい，自然言語で提供できる入力内容を基に有効な Pit を生成する手法を提案する．本手法を用いることにより，Peach によるファジング作業の効率向上が期待できる．

† 近畿大学 理工学部 情報学科
Department of Informatics, Faculty of Science and Engineering, Kindai University
‡ 近畿大学 情報学部 情報学科
Department of Informatics, Faculty of Informatics, Kindai University
§ 近畿大学情報学研究所
Cyber Informatics Research Institute, Kindai University

```
<?xml version="1.0" encoding="utf-8"?>
<Peach xmlns="http://peachfuzzer.com/2012/Peach" xmlns:xsi="http:/
    <DataModel name="HttpRequest">
        <String value="Hello World!" />
    </DataModel>

    <StateModel name="TheStateModel" initialState="TheState">
        <State name="TheState">
            <Action type="output">
                <DataModel ref="HttpRequest" />
            </Action>
        </State>
    </StateModel>

    <Agent name="RemoteAgent" location="tcp://192.168.1.190:9001">
```

図 1　Pit の例(抜粋)

2. 関連研究

LLM による生成内容には，誤情報や，古い情報，または知識不足が原因である「ハルシネーション」と呼ばれる有用でない内容が含まれることがある．そこで，Retrieval-Augmented Generation(以下，RAG) [5]と呼ばれる，生成タスクを実施する時に外部の知識源から情報を取得し，生成内容に反映することで出力精度を向上させる手法が知られている．また，プロンプトの設計を調整し，より望ましい出力を引き出すプロンプトエンジニアリングや，ファインチューニングと呼ばれる，学習済みのモデルを別のデータで再トレーニングして調整する手法も存在する．

Bassamzadeh らの研究では，ドメイン固有言語(以下，DSL)をコーディングする能力において RAG とファインチューニングを比較し，RAG がファインチューニングの品質に匹敵しており，有益であると結論付けている[6]．

本研究では LLM により Pit を生成する．この Pit はファジングテストの設定を記述するために特化した要素・構造を持っているため，記述には Peach の仕様を十分に理解する必要があり，これは特定のドメインに特化した DSL のようなものであるとみなすことができる．Pit のサンプルはインターネット上で様々なものが公開されているので，既存の LLM の学習データにも含まれている可能性が高い．しかし Peach には 2 系と 3 系があり，Pit の構造が異なり互換性がないため，希望するバージョンの Pit を生成するためには，適切な手段が必要となる．したがって，RAG またはファインチューニングなどを用いて出力精度を向上する必要がある．ファインチューニングは追加の学習に大量のデータセットや費用，時間を必要とするため，本研究においては RAG を採用した．

また，Brown らの研究では GPT に対するプロンプトエンジニアリングとして，One/Few-Shot learning の有効性について述べている[7]．Few-Shot とは，プロンプトにタスクに関する説明と少数の例を与えることで出力精度が向上する手法で，与える例の数が一つであった場合を特に One-Shot と呼ぶ．本研究においても，RAG により抽出された Pit サンプルを用いて One/Few-Shot learning を適用することで，出力精度の向上を図っている．

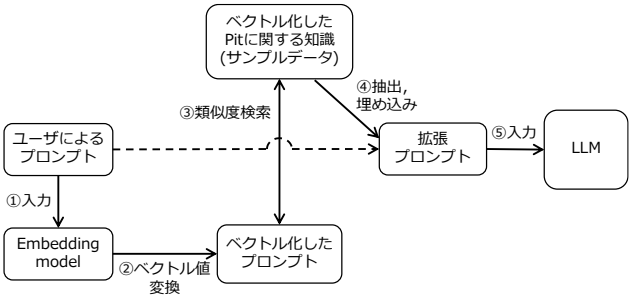

図 2　本研究における RAG の流れ

3.　研究内容

3.1　有効な Pit の生成手法

本研究では Peach のファジング設定支援に LLM と，その出力精度向上に RAG と Few-Shot learning を用いる．代表的な LLM として OpenAI 社によって開発された GPT シリーズがある．Web API 経由で利用可能な GPT の中で，現時点で特にコーディング能力，生成速度に優れている GPT-4 Turbo を使用する．GPT は，インターネット上の Web ページや書籍などから得た広範囲な情報を基に学習されており，C++や Java などの有名なプログラミング言語や，XML や JSON などの一般的なデータ形式を満足に扱うことができる．本研究で生成の対象とする Pit は，様々なサンプルが開発者や有志によってインターネット上に公開されている．しかし，LLM の学習に用いられる膨大なテキストデータの中で，その量は正確な生成を可能とするほど十分ではなく，GPT のみで有効な Pit を得るのは難しい．

そこで，Pit に関する専門的な知識を強化するために RAG を実装する．本研究における RAG の手順を以下と図 2 に示す．

① ユーザによるプロンプトを Web API 経由で OpenAI の Embedding モデルに入力
② Embedding モデルによってプロンプトをベクトル値に変換
③ 同様に Embedding モデルでベクトル化済みの Pit に関する専門的な知識(Pit のサンプルデータ)と，②で得たプロンプトのベクトル値のコサイン類似度を計算，比較
④ 類似度が高い上位のデータを抽出し，プロンプトに埋め込む
⑤ 拡張されたプロンプトを LLM(GPT-4 Turbo)に入力し，Pit を生成

①及び②ではユーザによる簡単な生成指示(プロンプト)を，OpenAI による Embedding(埋め込み)モデルである text-embedding-3-large を用いて 3072 次元のベクトル値に変換する．このようにテキストをベクトル化することで，単語や文の意味の特徴を数値化し，テキスト間の類似性や関連性を定量的に分析することが可能となる．また，同様に Pit に関する専門的な知識も Embedding モデルを用いてベクトル化する．本研究では Pit に関する専門的な知識として，Peach の開発者によって提供されている Pit のサンプル 24 個及び，自ら作成したサンプル 2 個を使用している．③ではプロンプトのベクトル値と，Pit サンプルデータのそれぞれのベクトル値を，コサイン類似度を用いて比較する．④では，コサイン類似度が高かった Pit サンプルを抽出する．今回，Pit の専門的な知識としてのサンプルデータが 26 個であり，それぞれのサンプルの想定ファジング状況も全く異なるため，類似度により抽出するサンプルデータの数は 1 個としている．⑤では，抽出した Pit サンプルをユーザが入力したプロンプトに追加，LLM へ入力し，Pit サンプルに基づいた出力を得ている．RAG によって抽出した Pit サンプルが追加されたプロンプトは，One-Shot learning が適用されている状態である．

```
入力:
Peach Fuzzer(バージョン3系)におけるHTTPリクエストのPitファイルを生成してください。なお、対象サーバのIPアドレスは192.168.56.20、ポート番号は8000とします。すべてXML形式のみで出力し、不要な文字列は含めないでください。XML形式として適切な改行を行ってください。
```

図 3　生成プロンプト(HTTP リクエスト)

3.2　生成結果

本手法により有効な Pit が生成されることを示すため，HTTP リクエスト，HTTP レスポンスの 2 通りに関して，ファジングを実行するための Pit の生成を試みる初期実験を実施した．ファジング作業者の負担軽減を目的とするため，単純かつ少ない文章量を意識し，図 3 に示すようなプロンプトを作成した．そして，本手法の適用前後に関係なく，LLM からの出力には冒頭や末尾に Pit 以外の余計な文字列が含まれていることがあるため，パーサを記述し自動で削除している．

本手法適用前の GPT-4 Turbo に対して図 3 に示すプロンプトを入力したところ，有効な Pit に類似した形式で記述されたテキストが出力された．しかし，出力結果に基づいて Peach を実行したところ，存在しないローカルエージェントを指定していたためエラーが発生し，ファジングを実行することはできなかった．

本手法適用後に同一のプロンプトを使用したところ，HTTP リクエスト，HTTP レスポンス共に，プロンプトによる要求を満たし，Peach で実行可能な出力を得た．生成及び Peach を実行するにあたり，使用したバージョンや一部パラメータを表 1 に列挙する．

表 1　生成及び実行環境の設定

項目	詳細
LLM	OpenAI GPT-4 Turbo (2024-04-09)
LLM temperature	0
Embedding model	OpenAI embedding-3-large
Peach	v3.1.124.0

4.　結論

本研究では，Peach を使用するファジング作業者の負担軽減を目的に，LLM を用いて，従来の Pit の記述に必要なプロトコルの詳細なデータ構造などに比べて作業者が理解しやすい，自然言語で提供できる入力内容を基に有効な Pit を生成する手法を提案した．本手法を用いることで，HTTP リクエスト，HTTP レスポンスに関して有効な出力が得られたことを確認した．今後，より多くのプロトコルやファジング状況に対応する Pit のサンプルを RAG の知識源として増やし，本手法の有効性を確認する予定である．

参考文献

[1] 総務省：令和 6 年版情報通信白書，入手先 <https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r06/pdf/00zentai.pdf>(参照 2024-07-14).

[2] Peach Fuzzer：Peach，入手先 <https://peachtech.gitlab.io/peach-fuzzer-community/>

[3] 情報処理推進機構：ファジング実践資料，入手先 <https://www.ipa.go.jp/security/vuln/fuzzing/ug65p9000001986g-att/000057628.pdf> (参照 2024-07-14).

[4] 永原 渓太郎，大野 堅太郎，吉田 則裕ほか：IoT の不具合に対するファジングツール Peach の有効性調査，情報処理学会研究報告， Vol.2022-SE-210, No.3, pp.1-8(2022).

[5] Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. : Retrievalaugmented generation for knowledge-intensive NLP tasks, Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS’20, Article 793, 9459-9474(2020).

[6] Nastaran Bassamzadeh, Chhaya Methani：A Comparative Study of DSL Code Generation: Fine-Tuning vs. Optimized Retrieval Augmentation, arXiv, arXiv:2407.02742[cs.SE](2024).

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, et al.：Language Models are Few-Shot Learners, arXiv, arXiv:2005.14165v4[cs.CL](2020)