LLMを使った
ソフトウェア脆弱性の自動注入システム

23G356 三枝 泰士 （最所研究室）

大規模言語モデル (LLM) を活用し、脆弱性を含む仮想マシンの作成を自動化するシステムを開発した。本システムは、脆弱性注入作業の一部を効率化する可能性を示したものの、実用には情報収集方法の多様化やプロンプト設計などさらなる工夫が必要であることが明らかになった。

1 はじめに

サイバー攻撃の増加に伴い、セキュリティ対策が重要視される一方で、人材不足が課題となっている。実践的なセキュリティ教育としてサイバー防御演習が有効だが、準備に手間と時間がかかる。特に、演習マシンへの脆弱性注入の自動化が進んでおらず、柔軟な演習の妨げとなっている。そこで、本研究ではLLMを活用し、脆弱性注入を自動化するシステム VuLLMakerを開発する。これにより、演習準備の負担を軽減し、効率的なセキュリティ教育を実現することを目指す。

2 課題

サイバー防御演習のライフサイクルには準備、ドライラン、実行、評価、反復の段階があり、特に準備フェーズには多大な時間と手間がかかる。脆弱性を持つ演習マシンの作成は準備フェーズの一部であり、現状では脆弱性の注入やテストの自動化が十分に進んでいない [1]。ドライランでは準備フェーズで発生した不具合を発見し修正を行うため、修正作業の発生を防ぐためにも攻撃コードの実行テストも行うことは重要である。よって、脆弱性を持つ演習マシンを作成するために、脆弱なバージョンの特定、適切な環境の構築、依存関係の解決、攻撃コードの実行テストを行う必要がある。しかし、このような作業は従来人間が行っており、人間向けの形式で情報が提供されることが多く、自動で作業を行うためには具体的な行動を決定する情報が不足する。LLMの活用により、自然言語からの情報抽出が期待されるが、誤った回答やサイバー攻撃につながる回答のフィルタリングによる制約が問題となる。

3 設計・実装

自動化システムの設計では、作業の自動実行だけでなく、自動実行に必要な作業の削減も重要であると考える。よって、 VuLLMakerへの入力を可能な限り単純にし、その情報から連鎖的に新たな情報を取得していくことで、バージョン情報の取得や仮想マシンの構築、攻撃の実行･検証をシステムに実行させる。LLMを活用するが、RAGを使わずにドキュメントを整理し直接入力する手法を採用し、プロンプト設計やモデル選定にも工夫を加えている。また、LLMの回答は間違うこともあるため、可能な部分は正規表現などで機械的に処理し、誤回答を防ぎつつLLMの実行コスト削減を両立する方針である。実装では、脆弱性を持つ仮想演習環境を自動構築するために、WebAPIやコンテナを利用して各機能を実行する。

- メインスクリプト: ユーザー入力を受け取り、脆弱性情報の収集、仮想マシンの作成、攻撃・防御の各フェーズを制御する。
- 脆弱バージョン検索: Metasploitのモジュールパス名をもとに、ディストリビューション情報推定 APIやパッケージ情報検索を通じて脆弱パッケージを特定する。
- 仮想インスタンス作成: Incusを用いて、攻

撃・防御マシンを自動構築し、古いディストリビューションのイメージを独自に用意して対応範囲を拡張する。

- 脆弱パッケージのインストールと設定: LLM を活用して、必要な設定を自動生成・実行し、脆弱性を発現させる環境を構築する。
- 攻撃・パッケージ追加: 攻撃マシンからリバースシェルを用いて防御マシンを侵害し、不足パッケージの追加や攻撃命令の発信を行う。

4 評価

実験では、Metasploit Framework[2] が提供する linux 用の local 環境用のモジュールすべてについて脆弱性の注入を試みた。その結果、成功したのは 2 件（約 2%） にとどまった。成功したケースはいずれも、パッケージのインストールのみで脆弱性が発現するものであり、追加の発現設定を含めた脆弱性注入の自動化には至らなかった。

失敗したケースのなかで、44 件（約 50%）では情報の取得に問題が生じた。特に、セキュリティパッチ情報の取得が不安定であった。この問題を解決するには、複数の情報源を活用し、確実にデータを取得できる仕組みを構築することが重要である。さらに、一部のモジュールには CVE が割り当てられておらず、それに関連するセキュリティパッチ情報を得ることができなかった。これに対処する新たなアプローチとして、モジュールのパス名を用いて Google 検索も活用し、有用な情報源を見つける方法が考えられる。

また、16 件（約 19%） は、LLM の回答が脆弱性の発現設定を行うには不完全であった。発現設定が不足しているモジュールに対し、 GPT-4o を用いても同様に不完全な回答が得られたことから、モデル変更以外のアプローチが必要である。そのためには、以下の対策が有効と考えられる。

- プロンプトの見直し
- 入力ドキュメント情報の整理
- LLM に脆弱性の発現設定手順の各ステップに着目させる
- LLM の回答の適切性を評価し、不適切な場合は再度問い合わせる

さらに、20 件（約 24%）では、実装の不具合が原因となった。特に、脆弱性が存在すると判定されたディストリビューションが古く、仮想マシンのイメージを用意できなかったケースが多かった。この問題への対策として、より多くのイメージを用意する方法が考えられるが、古い Incus 用の仮想マシンイメージのビルドには限界があるため、抜本的な解決策が求められる。

LLM によるディストリビューションの推定が不適切であったためにセキュリティパッチ情報の取得や仮想マシンイメージの選択に失敗したケースが全体の約 33%を占めていた。したがって、ディストリビューションの推定精度を向上させることが重要である。この問題に対しては、人手によるディストリビューションの推定作業を導入することも検討の価値がある。手動での作業負担は増加するが、エクスプロイトのターゲットとなるディストリビューションはドキュメントに記載されていることが多いため、大きな手間にはならないと見込まれる。ただし、これ以降の作業は、手間のかかる工程が多く含まれるため、人手に頼るべきではない。

参考文献

[1] 赤坂航平, 中村章人. ソフトウェア脆弱性の自動テストツール. 情報処理学会第 81 回全国大会, pp. 385–386, 2019.

[2] Rapid7. Github - rapid7/metasploit-framework: Metasploit framework. `https://github.com/rapid7/metasploit-framework`. [Online; accessed 10-Jan-2025].