텍스트 유사도 기반 RAG와 sLLM을 활용한 소프트웨어 보안 버그 리포트 템플릿 생성 및 예측 기법

손영준[O1] 양근석*[2]
[1]한경국립대학교 컴퓨터응용수학부, [2]한경국립대학교 컴퓨터응용수학부(컴퓨터시스템연구소)
dragondall2015@gmail.com, gsyang@hknu.ac.kr

Software Security Bug Report Template Generation and Prediction Method Using Text Similarity-Based RAG and a smaller LLM

Youngjun Son[O1] Geunseok Yang*[2]
[1]Department of Computer Applied Mathematics, Hankyong National University, [2]Department of Computer Applied Mathematics, Hankyong National University (Computer System Institute)

요 약

소프트웨어 유지보수 과정에서 보안 버그 리포트(SBR)는 보안 취약점을 신속히 식별하고 수정하는 데 중요한 역할을 한다. 그러나 리포터의 보안 지식 수준 차이로 인해 리포트 품질이 상이하며, 정보 누락 및 모호성이 발생할 가능성이 높다. 이를 해결하기 위해, 본 연구는 텍스트 유사도 기반 Retrieval-Augmented Generation (RAG)과 경량화된 거대 언어 모델(sLLM)을 활용하여 구조화된 SBR 템플릿을 생성하고, 보안 취약점 여부를 예측하는 기법을 제안한다. Kubernetes 기반 클러스터 환경에서 실험을 진행했으며, 텍스트 유사도를 사용해 유사 리포트를 분류하고, RAG를 통해 핵심 정보를 추출하며, sLLM을 통해 보안 리포트를 자동 생성하였다. 생성된 리포트는 설명 누락과 불명확성을 보완하며, 성능 평가 결과 기존 베이스라인 대비 높은 Precision(0.9756), Recall(0.9758), F1-Score(0.9757)을 기록했다. 본 연구는 1,000개 이상의 인스턴스를 기반으로 성능을 검증하였으며, 제안된 기법은 보안 리포트 품질과 예측 정확도를 개선하고, 보안 취약점의 신속한 식별을 지원한다.

1. 서 론

소프트웨어 유지보수에서 보안 버그 리포트(Security Bug Report, SBR)는 소프트웨어의 안정성과 신뢰성을 보장하기 위한 핵심 도구이다. 하지만 리포터(프로젝트 매니저, 개발자, 사용자 등)의 보안 지식 수준과 주관적인 해석 차이로 인해 리포트 품질이 일관되지 않고, 정보 누락 및 불명확성이 빈번하게 발생한다. 이러한 문제는 개발자가 보안 취약점을 신속히 식별하고 해결하는 데 방해 요소로 작용한다[1].

이러한 문제를 해결하기 위해, 텍스트 유사도 기반[2] RAG[2]와 경량화된 거대 언어 모델(sLLM)[2]을 결합한 기법을 제안한다. 본 기법은 비정형 데이터인 SBR을 구조화된 템플릿으로 전환하여 정보 누락과 모호성을 보완하고, 보안 취약점 여부를 정확히 예측하는 것을 목표로 한다. 이를 위해, 본 연구는 Kubernetes[4] 기반 클러스터 환경에서 실험을 진행하며, 텍스트 유사도 분석, 정보 추출, 템플릿 생성 및 예측 단계를 통해 RAG와 sLLM의 결합 효과를 평가하였다.

이 연구는 Ambari, Wicket, Derby, Camel 등의 오픈 소스 프로젝트 데이터[1]를 사용해 평가되었으며, 기존 연구 대비 SBR 품질과 예측 정확도를 크게 향상하였다.

2. 관련 연구

Ji et al.[1]은 보안 및 비보안 버그를 심각성과 비심각성으로 결합하여 특징을 분석하고, LSTM(Long Short-Term Memory) 모델을 활용해 보안 취약점 예측 성능을 향상시켰다. 이 연구는 평균 Precision 84%, Recall 87%, F1-Score 84%를 기록하며, 오픈 소스 프로젝트에서 기존 베이스라인 대비 성능 개선을 하였다.

Shu et al.[3]은 하이퍼파라미터 튜닝과 데이터 전처리가 보안 버그 리포트 분류에 미치는 영향을 분석하여 Recall을 44%에서 77%로 크게 향상시켰다. 이 연구는 데이터 정제와 하이퍼파라미터 최적화가 모델 성능에 미치는 중요성을 강조하였다.

본 연구는 기존 연구와 달리, 텍스트 유사도 기반 RAG와 sLLM을 결합하여 SBR을 자동 생성하고, 보안 여부를 예측하는 기법을 제안한다. RAG는 관련 데이터에서 핵심 정보를 추출하는 데 활용되며, sLLM은 이 정보를 구조화된 템플릿으로 변환하여 리포트의 일관성과 품질을 높인다. 이러한 접근은 기존의 분류 중심 연구보다 구조화된 템플릿 생성과 예측 정확성에서 뛰어난 성능을 보일 수 있다.

3. 본론

본 연구는 Kubernetes[4] 기반 클러스터 환경에서 텍스트 유사도 기반 RAG와 sLLM 모델을 결합하여 보안 버그 리포트를 자동으로 템플릿화하고, 이를 통해 보안 취약점 여부를 식별하는 기법을 제안한다. 제안된 방법론은 RAG를 통해 유사한 리포트를 식별하고, sLLM을 활용하여 구조화된 템플릿으로 재구성하며, 이 템플릿을 기반으로 보안 여부를 예측한다. 주요 단계는 아래와 같다.

1) sLLM을 위한 텍스트 전처리

보안 버그 리포트는 비정형 데이터로 구성되며, 코드 스니펫, 스택 트레이스, URL, 특수문자 등 불필요한 정보가 포함될 수 있다. 이러한 데이터를 모델이 효율적으로 처리할 수 있도록 전처리가 필요하다. 본 연구에서는 다음과 같은 전처리 과정[1]을 진행한다.

- **불필요한 기호 및 텍스트 제거**: 소스 코드, 스택 트레이스, URL 등 분석에 불필요한 기호 및 텍스트를 삭제
- **소문자화 및 불용어 처리**: 동일한 단어가 일관되게 처리될 수 있도록 소문자화하고, 분석에 크게 기여하지 않는 불용어를 제거
- **중요 정보 강조**: 보안과 관련된 주요 키워드를 중심으로 텍스트를 구성하여, 모델이 보안 핵심 요소에 집중하도록 설계. 이 과정에서 정보의 중복성을 최소화하여 템플릿의 간결성을 유지하고, 예측의 정확도를 향상시킴

[O]Speaker *Corresponding Author

2) 유사 보안 버그 리포트 식별

본 연구에서는 RAG와 코사인 유사도(Cosine Similarity)[2]를 활용하여 유사한 보안 버그 리포트를 식별한다. 코사인 유사도는 텍스트 벡터 간 각도를 계산하여 유사성을 측정하며, 각도가 작을수록 두 텍스트 간 높은 유사성을 의미한다. 이 과정은 다음과 같이 이루어진다

1. **FAISS(Facebook AI Similarity Search)[2]**: 대규모 텍스트 임베딩 벡터를 인덱싱하여, 새로운 보안 버그 리포트와 유사한 리포트를 신속히 검색
2. **텍스트 임베딩 변환**: 보안 버그 리포트를 임베딩 벡터로 변환하여, 데이터 간의 유사도를 분석
3. **유사성 기반 분류**: 동일 컴포넌트 내에서 유사한 보안 버그 리포트를 식별하고, 반복적으로 나타나는 취약점 패턴 및 구성 요소를 추출

3) sLLM을 활용한 보안 버그 리포트 템플릿 생성

RAG를 통해 추출된 정보를 바탕으로, sLLM(LLaMA2, 13B 파라미터[2])을 활용하여 구조화된 템플릿을 생성한다. 이 템플릿은 보안 리포트의 정보 누락을 보완하고, 명확하고 체계적인 구조를 제공한다. 본 연구에서 설계한 템플릿은 다음과 같은 주요 항목을 포함한다

- **Project Bug ID and Summary**: 리포트의 기본 식별 정보 및 요약
- **Revised Summary and Description**: 수정된 요약 및 상세 설명. 정보 누락을 보완하고 불명확성을 제거
- **Mitigation Steps**: 취약점을 해결하기 위한 임시 또는 최종 수정 방안
- **Similar Bug Information**: 유사 버그와의 연관성 정보 및 참고

이러한 템플릿은 생성된 SBR의 일관성을 유지하며, 보안 취약점을 신속히 파악할 수 있도록 돕는다.

〈페이지 제한으로 실제 결과를 일부만 요약하여 재구성함〉

[Summary and Description] - Summary: security wizard: ***webhcat*** server start fails on enabling security - Description: this happens when templeton.kerberos.principal property is set to http/_host@<***realm name***> instead of http/<***internal host name***e>@<realm name> **[Revised Summary and Description]** - Revised Summary: ***webhcat*** server start failure due to incorrect security realm configuration - Revised Description:. the issue occurs when the property is configured with `<***realm name***>` instead of using `<***internal host name***>` followed by `<realm name>`. …	**[Reproduction Step]** 1. enable the security wizard in ambari. 2. set the templeton… **[Expected and Actual Behavior]** - Expected Behavior : the webcat server shold… - Actual Behavior: the webhcat… **[Environment Information] …** **[Severity Level]…** **[Relevant Logs/Error Messages]** **…** **[Mitigation Steps]** - Temporary Solution**:** ensure that templeton.kerberos.principal… **[Similar Bug information]** - similar bugs ids: 3229 (0.55 similarity), … **[Similar Bug #3229 Summary]** ***webhcat*** server start failure due to incorrect security ***realm*** …

4) 제안 기법의 실행 환경

본 연구는 Kubernetes[4] 기반 클러스터에서 제안된 기법을 구현 및 실험하였다. Kubernetes 클러스터는 확장 가능하고 유연한 환경을 제공하여, 대규모 데이터 처리와 모델 학습을 효과적으로 지원한다. 모든 실험은 Kubernetes 클러스터 상에서 병렬로 실행되었으며, 성능 평가 및 데이터 관리의 효율성을 극대화하였다.

5) 의사코드

1. FUNCTION preprocess_text(report):
   a. cleaned_text ← REMOVE symbols from report
   b. lowercase_text ← CONVERT cleaned_text to lowercase
   c. keywords ← HIGHLIGHT keywords in lowercase_text
   d. RETURN keywords

2. FUNCTION find_similar_reports(reports, new_report):
   a. embeddings ← CONVERT reports to vector embeddings
   b. new_embedding ← CONVERT new_report to vector embedding
   c. similar_reports ← SEARCH for reports in embeddings that are similar to new_embedding with a threshold of 0.8
   d. RETURN similar_reports

3. FUNCTION generate_template(similar_reports):
   a. extracted_info ← RETRIEVE important information from similar_reports
   b. template ← GENERATE a template based on extracted_info
   c. RETURN template

4. FUNCTION predict_security(template):
   a. prediction ← PREDICT security-related outcome using template with a predefined prompt
   b. RETURN prediction

6) 기법의 장점

- **효율적인 데이터 처리**: FAISS를 활용해 대규모 데이터에서 유사 리포트를 빠르게 검색
- **구조화된 리포트 생성**: 템플릿 기반 접근법으로 정보 누락과 불명확성을 보완
- **보안 예측 정확도 향상**: sLLM을 활용하여 높은 Precision, Recall, F1-Score를 기록

실험 결과는 Ambari, Wicket, Derby, Camel 데이터셋에서 검증되었으며, 기존 연구 대비 높은 성능을 보였다

4. 실 험

1) 실험 구성

본 연구에서는 보안 버그 리포트 예측 성능을 평가하기 위해 4개의 오픈 소스 프로젝트[1].(Ambari : 2011-2014, Camel : 2007-2013, Derby : 2004-2014, Wicket: 2006-2014)의 데이터셋을 활용하였다.

실험은 Kubernetes 기반 클러스터 환경에서 수행되었으며, 이는 대규모 데이터 처리와 모델 학습의 효율성을 극대화하기 위함이다[4]. 모델 성능 평가는 머신러닝에서 일반적으로 사용하는 Precision, Recall, F1-Score를 기준으로 하며, 이진 분류(1=Security, 0=Non-Security)[1] 문제로 설정하였다.
각 프롬프트는 1회 질의(질문-답변)로 이루어졌으며, 모든 실험은 동일한 하이퍼파라미터를 적용하였다

2) 베이스라인 모델

Ji et al.[1]

- 보안 버그와 비보안 버그를 심각성과 비심각성으로 구분
- LSTM(Long Short-Term Memory) 모델을 적용하여 보안 여부를 예측
- 오픈 소스 프로젝트에서 Precision 83.75%, Recall 87.25%, F1-Score 83.75% 기록

Wu et al.[3]

- 레이블 정확성을 기반으로 보안 버그 리포트를 예측
- 5가지 머신러닝 기법(예: SVM, Decision Tree 등)을 비교하여 성능을 평가

- Precision 77.5%, Recall 43.75%, F1-Score 54.25% 기록

3) 실험 결과

표 1. 제안한 방법과 베이스라인간 성능비교

	Precision	Recall	F1-Score
Ji[1]	0.8375	0.8725	0.8375
Wu[3]	0.775	0.4375	0.5425
Our Model (LLaMA2)	0.9134	0.9067	0.9100
Our Model (RAG+LLaMA2)	0.9756	0.9758	0.9757

- Ji et al.[1] 모델은 Recall이 Precision보다 높음 오탐(False Positive)을 어느 정도 허용 전체적인 예측률을 높임. 그러나 F1-Score 0.8375로 여전히 일부 한계가 존재함
- Wu et al.[3] 모델은 Precision(0.7750)은 유지 Recall(0.4375)이 크게 낮아, 보안 버그 예측의 포괄성 측면에서 부족함. F1-Score는 0.5425로 가장 낮음
- Our Model (LLaMA2)는 베이스라인 모델(Ji et al., Wu et al.) 대비 전반적으로 높은 성능을 보임. Precision과 Recall 간의 균형이 유지되어 F1-Score 0.9100을 달성
- Our Model (RAG+LLaMA2)는 모든 모델 중 가장 우수한 성능을 보임. Precision, Recall, F1-Score에서 각각 97.56%, 97.58%, 97.57%로 가장 높은 결과를 보임. 이는 텍스트 유사도 기반 RAG가 관련 정보를 효과적으로 추출하고, sLLM이 이를 기반으로 일관된 템플릿을 생성한 결과로 해석됨

4) 제안 모델의 검증

본 연구는 다양한 데이터셋에 대해 모델의 일반화 가능성을 검증하였다. 특히, Kubernetes 클러스터 환경에서 병렬 처리를 통해 대규모 데이터 분석과 모델 학습을 수행함으로써 제안된 기법의 실효성을 보였다.

제안한 Our Model (RAG+LLaMA2)는 기존 베이스라인 모델 대비 높은 정확성과 예측 성능을 보였다. 이는 텍스트 유사도 기반 RAG와 sLLM의 결합이 보안 리포트의 품질과 예측 능력을 개선하는 데 효과적임을 보인다.

5. 토 의

본 논문은 텍스트 유사도 기반 RAG와 sLLM 모델을 결합하여 보안 버그 리포트 템플릿을 생성하고, 보안 여부를 정확히 예측하는 기법을 제안하였다. 비정형 데이터인 보안 버그 리포트를 구조화하여 정보 누락과 불명확성을 보완하고, 보안 취약점 식별 정확도를 크게 향상시켰다. 그러나 다음과 같은 한계와 향후 과제가 존재한다.

- **데이터 편향성**: Ambari, Camel, Derby, Wicket 데이터셋에서 실험했으나, 다른 오픈 소스 프로젝트에서도 성능 검증이 필요하며, 다양한 데이터셋으로 일반화 가능성을 평가할 예정이다.
- **보안 패턴의 다양성과 특이성**: 본 연구 템플릿이 고유 패턴을 완전히 반영하지 못할 가능성이 있어, 더 많은 보안 버그 리포트를 포함한 데이터셋을 수집하고 분석할 필요가 있다.
- **평가 지표의 확장**: Precision, Recall, F1-Score 외에 ROC-AUC, False Positive Rate(FPR) 등의 지표를 추가로 도입할 계획이다.
- **sLLM 모델의 한계**: 자원 효율성이 높지만, 대규모 LLM만큼 복잡한 문제를 다루지 못할 수 있어, 다양한 프로젝트와 데이터셋으로 성능 개선을 모색할 것이다.
- **하이퍼파라미터 최적화**: 기본 설정으로 진행한 실험을 최적화하여 성능 향상을 계획 중이다.

6. 결 론

본 연구는 텍스트 유사도 기반 RAG와 sLLM 모델을 활용하여 기존 보안 버그 리포트의 일관성과 품질 문제를 해결하고, 새로운 템플릿 기반 보안 버그 리포트 생성 및 보안 취약점 예측 기법을 제안하였다. Kubernetes 기반 클러스터 환경에서 실험한 결과, 제안된 모델은 Precision, Recall, F1-Score에서 기존 베이스라인 모델(Ji et al., Wu et al.) 대비 뛰어난 성능을 보였으며, 특히 RAG와 sLLM의 결합은 텍스트 데이터에서 핵심 정보를 효과적으로 추출하고 일관된 리포트를 생성하는 데엔 더 중요한 역할을 하였다

제안한 기법은 다음과 같은 주요 성과를 보였다.

1. 기존 연구 대비 보안 리포트 생성의 일관성과 예측 정확도 향상
2. 템플릿 기반 접근을 통해 정보 누락과 불명확성을 효과적으로 해결
3. 다양한 데이터셋에서 높은 성능을 기록하며, 범용적인 활용 가능성을 입증

향후 연구 방향

1. 보안 버그 리포트 데이터셋을 확장하여 다양한 보안 패턴과 특이성을 반영
2. 생성된 템플릿과 원본 리포트 간 정보 손실 및 의미 일치율을 정밀하게 검증
3. 추가적인 평가 지표를 도입하여 모델의 전반적인 성능을 다각도로 분석
4. 더 큰 LLM과 다양한 하이퍼파라미터 설정을 통해 성능을 최적화

본 연구는 보안 버그 리포트 생성 및 예측의 새로운 가능성을 열었으며, 소프트웨어 유지보수 과정에서 보안 취약점 식별의 정확성과 효율성을 높이는 데 기여할 것이다.

Acknowledgements

This research was supported by Hankyong National University Korea National University Development Project (2024)

참고문헌

[1]. J. Ji, G. Yang, “Enhancing Security Bug Report Prediction with Severity-Based Feature Selection through LSTM Algorithm,” in Proc. of the KCC, 2024.

[2]. Y. Gao, "Improving the Capabilities of Large Language Model Based Marketing Analytics Copilots With Semantic Search And Fine-Tuning," International Journal on Cybernetics & Informatics, 2024.

[3]. R. Shu, T. Xia, L. Williams, and T. Menzies, “Better Security Bug Report Classification via Hyperparameter Optimization,” arXiv preprint arXiv:1905.06872, 2019.

[4]. Wikipedia contributors, "Kubernetes," Wikipedia, accessed November 27, 2024, https://en.wikipedia.org/wiki/Kubernetes.