CVE-RAG: AI 코드 생성기의 취약 라이브러리 추천 방지를 위한 템플릿 프레임워크

유예지[1], 최현우[2]
[1]성신여자대학교 융합보안공학과 학사과정
[2]성신여자대학교 융합보안공학과 교수

esccy@naver.com, zemisolsol@sungshin.ac.kr

CVE-RAG: A Template Framework to Prevent Vulnerable Library Recommendations in AI Code Generators

Yeji Yu[1], Hyunwoo Choi[2]
1Department of Convergence Security Engineering, Sungshin Women's University
2Department of Convergence Security Engineering, Sungshin Women's University

요 약

최근 대규모 언어 모델(LLM) 기반의 AI 코드 생성 도구는 소프트웨어 개발 생산성을 크게 향상시켰지만, 학습 데이터의 시점 한계로 인해 최신 보안 취약점이 해결되지 않은 과거 버전의 라이브러리를 추천하는 문제가 있다. 본 논문에서는 최신 CVE 데이터베이스를 활용하는 RAG 시스템을 통해 코드 생성 전에 보안 템플릿을 생성하는 "CVE-RAG" 프레임워크를 제안한다. 이 프레임워크는 단순히 "최신 버전 사용" 지시보다 각 라이브러리에 대한 구체적인 취약점 정보를 기반으로 맞춤형 보안 가이드라인을 제공한다. 실험 결과, 제안된 프레임워크를 통해 생성된 코드는 템플릿 미적용 시에 비해 의존성 취약점이 100% 감소하였으며, 안전한 라이브러리 버전 사용률이 0%에서 100%로 향상되었다..

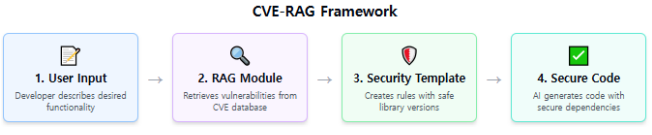
1. 서론

AI 코드 생성 도구는 개발자의 생산성을 혁신적으로 높이며 소프트웨어 개발 패러다임을 변화시키고 있다. GitHub Copilot, Cursor 등 다양한 도구들이 자연어 설명을 바탕으로 코드 스니펫, 함수, 심지어 전체 애플리케이션 구조까지 생성해준다. 그러나 이러한 도구들은 대규모 데이터셋으로 사전 학습되는데, 이 데이터에는 과거의 프로그래밍 패턴과 특정 시점까지의 라이브러리 정보가 포함되어 있다[1]. 이로 인해 모델은 최신 보안 패치가 적용되지 않은, 즉 알려진 취약점을 가진 라이브러리 버전을 코드 예시나 의존성 목록에 포함하여 추천하는 경향이 있다. 본 연구에서는 이러한 문제점을 해결하기 위해, 코드 생성 전에 최신 보안 정보를 반영한 가이드라인을 AI에게 제공하는 선제적 접근법을 제안한다. 단순히 "최신 버전을 사용하라"는 지시와 달리, 본 연구의 CVE-RAG는 CVE 데이터베이스에서 특정 라이브러리의 취약점 정보를 검색하여 맞춤형 보안 가이드라인을 제공한다. 최신 CVE 데이터베이스에 RAG 기술을 적용하여 AI 코드 생성 도구가 사용할 보안 템플릿을 동적으로 생성하는 CVE-RAG 프레임워크를 개발하였다.

2. 관련 연구

LLM 기반 코드 생성의 보안 취약점을 해결하기 위한 연구는 주로 생성된 코드의 사후 검증 및 개선에 초점을 맞춰왔다. Vul-RAG[1]는 기존 CVE 데이터를 RAG 시스템과 결합하여 LLM이 생성한 코드의 잠재적 취약점을 탐지하는 프레임워크를 제안했다. SOSecure[2]는 StackOverflow 데이터를 RAG에 활용하여 생성된 코드의 보안성을 평가하는 시스템을 소개했다. Codexity[3]는 정적 분석 도구의 결과를 피드백으로 활용하여 LLM이 보안 취약점을 포함한 코드를 수정하도록 유도하는 프레임워크를 제시했다. 이러한 선행 연구들은 모두 코드 생성 이후에 보안 문제를 탐지하는 '사후 대응적' 접근 방식인 반면, 본 연구에서 제안하는 CVE-RAG 프레임워크는 코드 생성 전에 보안 정보 기반의 템플릿을 제공하는 '선제적 예방' 접근 방식을 취한다.

3. 제안하는 프레임워크



(그림 1) CVE-RAG 프레임워크 개념도

본 논문에서 제안하는 CVE-RAG 프레임워크는 그림 1과 같이 크게 RAG 모듈, 템플릿 생성기, AI 도구 인터페이스로 구성된다.
사용자 입력: 개발자는 자연어로 구현하고자 하는

기능이나 사용할 라이브러리에 대한 설명을 제공한다.

RAG 모듈: 사용자 입력을 바탕으로 관련성 높은 취약점 정보를 CVE 데이터베이스에서 검색한다. 본 연구에서는 로컬 JSON 데이터베이스와 TF-IDF/코사인 유사도 검색을 사용하였다.

템플릿 생성기: 검색된 취약점 정보를 분석하여, AI 코드 생성 도구가 이해할 수 있는 형식의 보안 템플릿을 생성한다. 본 연구에서는 Cursor AI가 사용하는 .cursorrules 형식의 템플릿 생성을 목표로 하였다. 이 템플릿에는 다음 정보가 포함된다: 안전한 라이브러리 버전 규칙(특정 라이브러리에 대해 알려진 취약점을 회피하는 최소 버전을 명시하는 규칙, 예: requests(?!>=2.32.0))과 취약한 코드 패턴 규칙(특정 CVE와 관련된 잠재적으로 취약한 코드 작성 패턴을 탐지하는 규칙, 정규식 기반).

AI 도구 인터페이스: 생성된 템플릿을 AI 코드 생성 도구의 작업 환경에 적용한다. AI 도구는 이 템플릿을 참조하여 코드 생성 시 안전한 버전을 우선 제안하고, 취약한 패턴 작성 시 사용자에게 경고를 표시하게 된다.

CVE-RAG는 "최신 버전 사용"과 같은 일반적인 지시와 달리, 특정 라이브러리의 특정 취약점에 대응하는 최소 안전 버전을 명확히 식별함으로써 보다 정확하고 맞춤화된 보안 가이드라인을 제공한다.

4. 실험 설계

제안된 CVE-RAG 프레임워크의 효과를 검증하기 위해 다음과 같이 실험을 설계하였다. 실험 목표는 CVE-RAG 템플릿 적용 유무에 따른 AI 생성 코드의 보안성 비교이며, 실험 환경으로는 Cursor AI 도구를 사용하여 CVE-RAG 생성 .cursorrules 적용 그룹(A)과 미적용 그룹(B)으로 나누어 비교하였다. 실험 프롬프트로는 이미지 처리(Flask와 Pillow 사용), API 통신(requests 사용), 데이터베이스 연동(Django 사용) 세 가지 시나리오를 사용했으며, 측정 지표는 pip-audit를 사용한 의존성 취약점 분석과 Bandit을 사용한 코드 패턴 취약점 분석을 진행하였다. 각 실험 조건별로 여러 번 반복 실험을 수행하고 결과의 평균을 계산하여 신뢰성을 높였다.

5. 실험 결과 및 분석

Prompt	Condition	Total Dependencies	Vulnerabilities	Major Vulnerable Libraries
Image Processing	Without CVE-RAG	10	9	Pillow 10.0.0, Werkzeug 2.3.7
Image Processing	With CVE-RAG	10	0	None (Pillow 11.1.0, Werkzeug 3.1.3)
API Communication	Without CVE-RAG	7	1	Requests 2.31.0
API Communication	With CVE-RAG	6	0	None (Requests 2.32.3)
Database Operation	Without CVE-RAG	5	4	Cryptography 41.0.7
Database Operation	With CVE-RAG	4	0	None (Cryptography 44.0.2)
Average	**Without CVE-RAG**	**7.3**	**4.7**	-
Average	**With CVE-RAG**	**6.7**	**0**	-

<표 1> CVE-RAG 템플릿 적용 유무에 따른 프롬프트별 취약점 수 비교

pip-audit 분석 결과, 그룹 A(템플릿 적용)는 그룹 B(템플릿 미적용)에 비해 의존성 취약점이 100% 감소하였다. 구체적으로: 이미지 처리 프롬프트에서는 Pillow 10.0.0의 5개 취약점과 Werkzeug 2.3.7의 4개 취약점이 모두 제거되었고, API 통신 프롬프트에서는 Requests 2.31.0의 CVE-2024-35195 취약점이 제거되었으며, 데이터베이스 프롬프트에서는 Cryptography 41.0.7의 4개 취약점이 모두 제거되었다. 이는 CVE-RAG 템플릿의 버전 제약 조건이 AI가 안전한 라이브러리 버전을 추천하도록 효과적으로 유도했음을 보여준다.

또한, CVE-RAG 프레임워크가 생성한 .cursorrules 템플릿을 분석한 결과, 각 프롬프트마다 관련 라이브러리의 최신 취약점 정보를 정확히 반영한 규칙이 생성된 것을 확인할 수 있었다.


```
{
  "version": 1,
  "rules": [
    {
      "name": "Use safe version of Pillow",
      "pattern": "pillow(?!>=10.3.0)",
      "message": "Security: Use Pillow>=10.3.0 to avoid CVE-2024-28219"
    },
    {
      "name": "Use safe version of requests",
      "pattern": "requests(?!>=2.32.0)",
      "message": "Security: Use requests>=2.32.0 to avoid CVE-2024-35195"
    },
    {
      "name": "Use safe version of cryptography",
      "pattern": "cryptography(?!>=42.0.4)",
      "message": "Security: Use cryptography>=42.0.4 to avoid CVE-2024-26130"
    }
  ]
}
```


(그림 2) 보안 규칙 템플릿

템플릿은 프롬프트에서 사용된 핵심 라이브러리에 대해 각각의 CVE에 대응하는 최소 안전 버전을 명시하며, 규칙은 정규식 패턴으로 안전하지 않은 버전 사용을 탐지하고, 구체적인 CVE ID를 포함한 메시지를 제공하여 개발자에게 정확한 보안 정보를 전달한다.

6. 결론

본 논문에서는 AI 코드 생성 도구가 취약한 라이브러리 버전을 추천하는 문제를 해결하기 위한 CVE-RAG 프레임워크를 제안하였다. 세 가지 프롬프트에 대한 실험 결과, 제안된 프레임워크는 AI가 생성하는 코드의 의존성 취약점을 평균 100% 감소시켰으며(14개 → 0개), 안전한 라이브러리 버전 사용률을 0%에서 100%로 향상시키는 효과를 보였다. CVE-RAG는 단순히 "최신 버전 사용"이라는 일반적인 지시와 달리, 특정 라이브러리의 구체적인 취약점 정보를 기반으로 맞춤형 보안 가이드라인을 제공한다는 점에서 차별화된 가치를 가진다. 이는 코드 생성 전에 최신 보안 정보를 반영한 가이드라인을 제공하는 선제적 접근 방식이 AI 기반 소프트웨어 개발의 보안성을 강화하는 데 효과적임을 보여준다. 향후 연구에서는 다양한 AI 코드 생성 도구에 적용 가능한 범용 템플릿 형식 개발과 함께, 코드 패턴 취약점 감지 능력을 향상시키는 방향으로 프레임워크를 발전시킬 계획이다.

참고문헌

[1] Zhang, Y., et al. "Vul-RAG: A Vulnerability Detection Framework based on Retrieval-Augmented Generation for Large Language Models," arXiv preprint arXiv:2406.11147, 2024.

[2] Siddiq, M. L., et al. "SOSecure: A Knowledge-enhanced RAG System for Secure Code Generation by LLMs," arXiv preprint arXiv:2403.13654, 2024.

[3] Wu, H., et al. "Codexity: A Framework for Mitigating Security Vulnerabilities in LLM-Generated Code via Static Analysis Feedback," arXiv preprint arXiv:2405.03927, 2024.