패키지 내부 메타데이터를 활용한 RubyGems 악성 패키지 탐지

임규보 [1], 박상준 [2], 김미수 [3]
[1] 전남대학교 지역시스템바이오공학과 학부생
[2] 전남대학교 컴퓨터정보통신공학과 학부생
[3] 전남대학교 인공지능융합학과 교수

[1]98limgbo@gmail.com, [2]191416@jnu.ac.kr [3]misoo.kim@jnu.ac.kr

RubyGems Malicious Package Detection Using Internal Metadata of Package

Gyu-Bo Lim[1], Sang-Joon Park[2], Mi-Soo Kim[3]
[1]Dept. of Rural and Biosystems Engineering, Chonnam University
[2]Dept. of Computer Engineering, Chonnam University
[3]Dept. of Artificial Intelligence Convergence, Chonnam University

요 약

오픈 소스 소프트웨어 공급망의 성장과 함께 악성 패키지 탐지가 중요한 과제가 되고 있다. 기존의 정적 분석과 동적 분석은 시간이 많이 소요되기 때문에, 본 연구에서는 이를 해결하기 위해 메타데이터 분석을 도입했다. 또한, 외부 메타데이터에 의존하는 기존 탐지 모델의 한계를 극복하고자, 내부 메타데이터만을 활용하여 배포 초기에도 RubyGems 에서 악성 패키지를 신속하게 탐지할 수 있는 모델을 제안한다. 실험 결과, 제안한 모델은 평균 97.63%의 정확도로 우수한 성능과 빠른 탐지 속도를 보여, 내부 메타데이터 기반의 탐지 방식의 가능성을 확인하였다.

1. 서론

오픈 소스 소프트웨어 공급망의 급속한 성장으로 악성 패키지에 의한 보안 문제가 부각되고 있어, 이를 신속하게 탐지하는 것이 필요하다[1]. NPM 과 PyPI 같은 주요 공급망에서는 악성 패키지 탐지 연구가 활발히 진행되고 있지만, RubyGems 에서는 이러한 연구가 상대적으로 부족하다[2]. RubyGems 은 루비 언어를 기반으로 한 패키지 공급망으로, 웹 애플리케이션과 서버 개발에 중요한 역할을 하고 있어, 이를 사용하는 프로젝트와 서비스가 많다. 따라서, RubyGems 내 악성 패키지 탐지에 대한 연구가 필요하다.

기존 악성 패키지 탐지 방법으로는 정적 분석과 동적 분석이 주로 사용되나, 이는 많은 시간과 비용이 소요되어 신속한 탐지에 어려움이 있다[3]. 이를 해결하기 위해 패키지 메타데이터를 활용하여 악성 패키지를 탐지하는 MeMPtec 모델이 제안되었다[4]. 그러나 이 모델은 NPM 공급망에 초점을 맞추며, 패키지 다운로드 수나 포크 수와 같은 외부 메타데이터에 의존하고 있어 배포 초기에 탐지하는 데 어려움이 있다. 이에 본 연구에서는 배포 초기에 악성 패키지를 탐지하기 위해, 패키지 내부 메타데이터만을 활용하여 RubyGems 에서 배포된 악성 패키지를 신속하게 탐지할 수 있는 모델을 제안한다.

2. 관련 연구

MeMPtec 은 NPM 패키지 공급망에서 내부 메타데이터(패키지 이름, 설명, 의존성 등)와 외부 메타데이터(다운로드 수, 포크 수 등)를 활용하여 악성 패키지를 탐지하는 모델이다. 다양한 머신러닝 알고리즘으로 학습하며, 특히 DRF 알고리즘에서 높은 성능을 보였다. 그러나 MeMPtec 은 외부 메타데이터에 의존하는 한계가 있다. 외부 메타데이터는 해당 패키지에 대한 사용자 상호작용으로 수집된다. 따라서, 사용자 상호작용이 적은 배포 초기에는 외부 메타데이터가 충분히 축적되지 않기 때문에, 악성 패키지를 초기에 탐지하기 어렵다. 이로 인해, 사용자들이 잠재적인 보안 위협에 노출될 수 있다.

3. 모델 개요 및 데이터 처리

3.1 모델 개요

본 연구에서는 RubyGems 패키지 공급망에서 악성 패키지를 탐지하기 위해 패키지 내부 메타데이터만을

이용한 모델을 제안한다. 내부 메타데이터는 패키지 이름, 버전 정보, 작성자 정보, 의존성 정보 등 총 28 개의 항목으로 구성되어 있으며, 이를 활용하여 SVM, GLM, GBM, DRF 알고리즘을 적용한 탐지 모델을 구축하였다.

3.2 학습 데이터 수집

악성 패키지는 Backstabber's Knife Collection 815 개, MalOSS 9 개, snyk 2 개로 총 826 개를 수집하였고[5], 정상 패키지는 다운로드 수가 많은 상위 패키지를 선정하여 총 826 개를 수집하였다[6].

3.3 메타데이터 구성 및 전처리

패키지에서 추출할 수 있는 내부 메타데이터는 총 28 개로 구성되었으며, <표 1>에 정리하였다. 이를 모델에 적용하기 위해 전처리 과정을 수행하였다. 예를 들어, 패키지 이름의 길이를 값으로 변환하고, 날짜는 연/월/일로 분할하였으며, 버전 정보는 각 구성 요소로 분할하였다. 또한, 파일 및 의존성 목록의 개수를 계산하고, 인증서 체인과 같은 일부 항목은 존재 여부에 따라 One-Hot 인코딩을 적용하였다.

<표 1> 내부 메타데이터 주요 구성 항목

분류	항목
패키지 이름	패키지 이름
버전 정보	패키지 버전, 패키지 구동에 필요한 최소 Ruby 버전, 패키지 설치에 필요한 최소 RubyGems 버전, 패키지 생성 시 사용된 RubyGems 버전, 패키지 메타데이터 형식 버전
패키지 설명	패키지 간단한 요약, 패키지 자세한 설명, 설치 후 출력 메시지
파일 정보	패키지 내 파일 목록, 테스트 파일 목록, 패키지에 포함된 실행 파일 목록, 패키지 사용 시 참조할 경로, 패키지에 포함된 확장 기능 목록, 패키지 사용 시 자동으로 요구할 파일, 문서화를 위한 추가 파일 목록, 문서화 도구에 적용할 옵션
참여자 정보	패키지 작성자, 작성자의 이메일 주소
배포 정보	패키지 릴리스 날짜
의존성 정보	패키지가 의존하는 다른 패키지 목록, 동작 시 필요한 패키지 목록, 개발 및 테스트 환경에서 필요한 패키지 목록
링크 정보	패키지 공식 홈페이지 링크, 개발 관련 URI 정보
플랫폼 정보	패키지가 작동하는 플랫폼
보안 정보	패키지 서명에 사용된 키, 인증서 체인
기타 정보	패키지 라이선스 정보, 패키지를 사용하기 위한 추가 요구사항

4. 실험 및 결과

본 모델을 SVM, GLM, GBM, DRF 알고리즘으로 학습하고, Accuracy, Precision, Recall, F1-Score 로 성능을 평가하였다(결과는 <표 2> 참조). GBM 과 DRF 알고리즘은 각각 Accuracy 0.9861, 0.9879 및 F1-Score 0.9860, 0.9877 로 높은 탐지 성능을 보였다. 이는 외부 메타데이터 없이도 높은 정확도와 탐지 성능을 달성하였음을 보여준다. 반면, SVM 과 GLM 알고리즘은 상대적으로 낮은 성능을 보였으나, 의미 있는 탐지 성능을 유지하였다.

<표 2> 머신러닝 알고리즘별 모델 성능 평가 결과

Model / Metric	Accuracy	F1-Score
SVM	0.9710 ± 0.0062	0.9707 ± 0.0061
GLM	0.9601 ± 0.0115	0.9603 ± 0.0112
GBM	0.9861 ± 0.0059	0.9860 ± 0.0061
DRF	0.9879 ± 0.0081	0.9877 ± 0.0083

5. 결론

본 연구에서는 내부 메타데이터만을 활용한 악성 패키지 탐지 모델을 제안하였다. 실험 결과, GBM 과 DRF 알고리즘을 적용한 모델이 우수한 탐지 성능을 보여주었으며, 이를 통해 패키지 배포 초기 단계에서도 선제적 탐지가 가능할 것으로 기대된다. 앞으로의 연구에서는 Ruby 뿐만 아니라 다양한 프로그래밍 언어의 메타데이터를 활용하여 모델을 확장하고, 하나의 모델로 다중 언어에서 악성 패키지를 탐지할 수 있는 방법을 모색할 계획이다.

사사

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 인공지능융합혁신인재양성사업(IITP-2023-RS-2023-00256629), 대학 ICT 연구센터사업(IITP-2024-RS-2024-00437718), 정보통신기획평가원의 지원(No.RS-2024-00438686, 비정상 오픈소스 식별 및 DevSecOps 자동 적용을 통한 소프트웨어 신뢰성 향상 기술 개발)을 받아 수행된 연구임.

참고문헌

[1] Wentao Liang, Xiang Ling, Jingzheng Wu, Tianyue Luo, and Yanjun Wu, "A Needle is an Outlier in a Haystack: Hunting Malicious PyPI Packages with Code Clustering," Proc. IEEE/ACM International Conference on Automated Software Engineering (ASE), Kirchberg, Luxembourg, Sep. 2023, pp. 307-318.

[2] Ahmed Zerouali, Tom Mens, Alexandre Decan, and Coen De Roover, "On the Impact of Security Vulnerabilities in the npm and RubyGems Dependency Networks," Empirical Software Engineering, vol. 27, no. 5, pp. 1-45, 2022.

[3] Muhammad Ijaz, Muhammad Hanif Durad, and Maliha Ismail, "Static and Dynamic Malware Analysis Using Machine Learning," Proc. of the 2019 16th International Bhurban Conference on Applied Sciences & Technology (IBCAST), Islamabad, Pakistan, Jan. 2019, pp. 687-691.

[4] Sajal Halder, Michael Bewong, Arash Mahboubi, Yinhao Jiang, Md Rafiqul Islam, Md Zahid Islam, Ryan HL Ip, Muhammad Ejaz Ahmed, Gowri Sankar Ramachandran, Muhammad Ali Babar. "Malicious Package Detection using Metadata Information." roc. of the ACM Web Conference 2024, Singapore, May 2024, pp. 1779-1789.

[5] Xiaoyan Zhou, Ying Zhang, Wenjia Niu, Jiqiang Liu, Haining Wang, and Qiang Li, "OSS Malicious Package Analysis in the Wild," arXiv preprint, 2024.

[6] Xiaoyan Zhou, Feiran Liang, Zhaojie Xie, Yang Lan, Wenjia Niu, Jiqiang Liu, and Qiang Li, "A Large-scale Fine-grained Analysis of Packages in Open-Source Software Ecosystems," arXiv preprint, 2024.