한국융합보안학회 2026년 하계학술대회 논문집

업데이트 기반 확장 프로그램 공급망 보안 시스템 설계

정재성*, 심재훈, 박혜수, 윤주혁, 곽민경, 최한림, 임정묵, 이병천
중부대학교

Design of a Supply Chain Security System for Update-Based Browser Extensions

Jaeseong Jung*, Jaehun Sim, Hyesu Park, Juhyeok Yun, Minkyung Kawk,
Hanrim Choi, Jeongmook Lim, Byoungcheon Lee

브라우저 확장 프로그램을 공급망 보안 관점에서 재정의하고, 유입·검증·저장·재평가로 이어지는 순환 구조의 다층 검증 체계를 제안한다. 버전·권한·외부 통신·평판 변화를 지속적으로 추적하고 정적·동적 분석과 평판 정보를 결합하며, RAG 기반 시나리오 매칭과 LLM을 통해 관리자의 승인·차단 판단을 지원함으로써 단일 시점 탐지가 아닌 지속 운영 가능한 공급망 보안 모델을 구현한다. 이 구조는 VS Code Extension, 플러그인 및 데스크톱 앱 등 외부 저장소·자동 업데이트를 공유하는 다양한 소프트웨어 공급망으로 확장 가능하다.

Key Words : Supply Chain Security(공급망 보안), Browser Extension(브라우저 확장 프로그램), Update-based(업데이트 기반), Static Analysis(정적 분석), Dynamic Analysis(동적 분석)

1. 서 론

사이버 보안 분야에서 공급망 공격은 특정 개발 생태계에 국한되지 않고, 소프트웨어의 생산·유통·유지 과정 전반에서 발생할 수 있는 보안 위협이다[1]. 최근 브라우저 확장 프로그램을 대상으로 한 공급망 공격이 증가하고 있으며, 사용자 편의를 위해 부여된 높은 권한이 민감 정보 탈취로 악용되는 사례가 늘어나고 있다[2]. 확장 프로그램은 공식 스토어를 통해 설치된 후 자동으로 업데이트되는 구조를 가지는데, 공격자가 초기에 정상 기능을 제공하여 신뢰를 확보한 후 업데이트 과정에서 데이터 수집이나 외부 통신 기능을 추가하는 공급망 공격 경로로 악용될 수 있다.

1) 문제 제기

Manifest V3 도입, 원격 코드 실행 차단, 등록·업데이트 정책 위반 검토 등 Chrome 공식 스토어와 브라우저 제조사 측의 대응이 있지만, 정책 심사 기준은 비공개이며 설치 이후 버전 변경·외부 서버 응답 변화 등 동적 행위를 지속적으로 검증하기에는 한계가 있다[3]. 심지어 manifest.json으로부터 정상·악성 확장 프로그램 둘 다 동일한 권한 구성을 사용하는 구조적 한계마저 존재한다[4]. 또한 사용자는 스토어 등록 여부, 평점, 리뷰, 설치 수와 같은 정보를 안전성의 지표로 활용하지만, 악성 확장 프로그램이 이러한 검증을 우회하여 신뢰를 유지하기에, 이러한 지표만으로 보안성을 보장하기는 어렵다.

2) 연구 목표

본 연구는 확장 프로그램을 단일 시점에서 분석하는 것이 아니라, 버전·평판 변화를 지속적으로 관리해야 하는 공급망 자산으로 보고, 유입·검증·저장·재평가로 이어지는 순환 구조의 다층 검증 체계를 제안한다.

1.1 연구 배경

본 시스템은 확장 프로그램 수집·배포를 위한 백엔드로 Python 기반 FastAPI와 Uvicorn을 사용하고, Nexus Raw Repository의 REST API와 연동하여 확장 프로그램 아카이브를 내부 저장소로 유입·관리한다. 분석 엔진은 오픈소스 ExtAnalysis와 VirusTotal, ClamAV를 통합해 정적·동적 분석을 수행하며, 정적 분석에서 수집한 메타데이터는 RAG 기반 구조에 활용된다. 이 과정에서 BGE-M3 임베딩 모델로 분석 결과를 벡터화하고 PostgreSQL 기반 Vector DB에 저장된 악성 행위 시나리오와의 유사도를 평가하며, LLM이 최종 위험 판단을 보조한다. 또한 Chrome 공식 스토어 정보와 LayerX 검사 보고서를 활용해 버전 및 평판 변화를 주기적으로 확인하는 'Retro 분석'을 수행하며, 전체 서비스는 AWS 클라우드 환경에 배포되어 있고 향후 Docker 기반 배포로 운영 편의성과 확장성을 높이는 것을 목표로 한다.

2. 본 론

본 연구에서 제안하는 시스템은 확장 프로그램을 단일 시점에서 검사하는 방식이 아니라, 유입·검증·저장·재평가가 반복되는 공급망 보안 구조로 설계된다.
본 장에서는 확장 프로그램 공급망에서 발생할 수 있는 피해 사례를 분석하고, 이를 완화하기 위한 연구 방법론을 제시한다. 이후 버전 추적과 대기열 운영이 포함된 순환 구조의 다층 검증 체계를 다룬다.

2.1 확장 프로그램 공급망 피해 사례 분석

확장 프로그램 공급망 공격은 처음부터 명확한 악성 프로그램을 배포하는 방식에 한정되지 않고, 정상·악성 버전이 번갈아 배포되는 방식으로 발생한다. 이때 사용자는 버전별 변화 사실을 인지하기 어렵기 때문에, 단일 시점 검사만으로는 공격을 식별하기 어렵다.

1) 피해 사례

GitLab Threat Intelligence가 보고한 악성 Chrome 확장 프로그램 사례에서는 다수의 확장 프로그램이 사용자의 브라우저 행위에 직접 영향을 미치는 기능을 수행한 것으로 나타났으며, ShadyPanda 캠페인 역시 장기간 정상 운영되던 확장 프로그램이 업데이트 과정에서 악성화되어 사용자 환경에 광범위한 영향을 미친 사례로 보고된다. 이러한 사례들은 확장 프로그램이 공식 스토어에 등록되어 있거나 높은 설치 수와 평판을 보유하더라도, 이후 배포되는 버전까지 동일한 수준의 안전성을 보장하지 못함을 보여준다[5].

2) 악성화 요인

개발자가 의도적으로 기능을 변경하여 확장 프로그램을 악용할 수도 있고, 계정이나 배포 권한이 탈취되어 정상 확장 프로그램이 변조될 수도 있다. 또한 외부 서버에서 설정값이나 스크립트를 받아 동작하는 구조에서는 확장 프로그램 자체 코드가 변경되지 않더라도, 서버 응답의 변화에 따라 실제 동작이 달라질 수 있다. 따라서 공급망 보안에서는 단일 시점의 검사만으로는 업데이트를 통한 위험을 줄이기 어렵기 때문에, 악성 여부를 단정하기보다 버전, 권한, 외부 통신, 평판 등의 변화를 지속적으로 관찰하는 것이 중요하다[1][3].

2.2 연구 방법론

본 연구는 이러한 피해 사례를 바탕으로, 확장 프로그램을 공급망 자산으로 다루기 위한 방법론을 제시한다[1][3].

1) 대기열 운영

개발 패키지 공급망 보안에서 활용되는 대기열 운영 개념을 확장 프로그램 환경에 적용한다. Nexus IQ 및 Lifecycle 계열 도구는 개발 패키지의 취약점, 정책 위반 여부를 검사하고, 신규·변경 패키지를 일정 기간 보류하면서 평판 정보를 확보하는 방식으로 공급망 위험을 줄인다. 이에 따라, 신규 또는 업데이트된 확장 프로그램을 즉시 배포하지 않고 대기열에 보관한 뒤 버전·평판 분석 결과를 종합적으로 검토하도록 설계한다.

2) 버전 추적

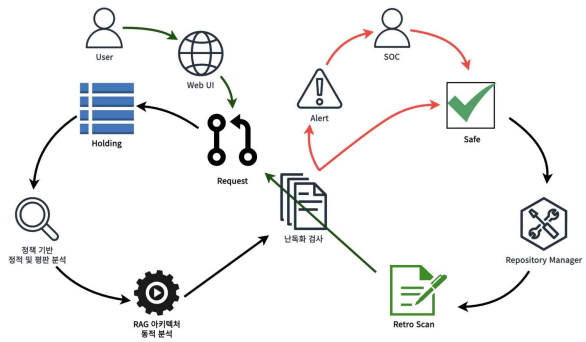
버전 추적은 본 방법론의 핵심 구성 요소이다. 확장 프로그램의 기존 버전과 신규 버전을 비교하여 manifest.json의 권한, 서비스 워커 또는 백그라운드 진입점, 외부 통신 대상의 변화를 확인한다[4]. 예를 들어 이전 버전에는 존재하지 않던 광범위한 호스트 접근 권한이 추가되거나, 외부 통신 대상이 변경되거나, 백그라운드 실행 흐름이 새롭게 추가되는 경우 이를 위험 징후로 기록한다. 또한 과거에 승인된 확장 프로그램이라도 새로운 버전이 배포되거나 평판 정보가 변하면 재평가 대상으로 편입된다.

3) 분석

정적·동적 분석은 이러한 순환 구조를 보조하는 기능으로 활용된다[2]. 분석 결과는 확장 프로그램을 악성으로 단정하기 위한 근거보다는, 취약점 존재나 위험 행위 가능성을 판단하기 위한 정보로 사용된다. 개별 지표(예: 외부 네트워크 요청, 동적 실행 API 등)를 단독으로 존재한다는 이유만으로 위험이라 간주하기보다, 어떤 행위 조합을 형성하는지, 기존 공격 시나리오와 유사한지, 이전 버전과 비교해 어떤 변화가 있었는지를 함께 고려해 관리자의 판단을 지원한다.

2.3 시스템 설계

본 연구에서 제안하는 체계는 (그림 1)과 같이 “유입(요청·수집)→검증(대기열·분석·승인/보류)→저장→재평가”의 순환 구조로 동작한다.



(그림 1). 확장 프로그램 공급망 보안을 위한 순환 구조의 다층 검증 체계

1) 유입

사용자는 외부 스토어에서 직접 확장 프로그램을 설치하지 않고, 내부 서비스를 통해 검색 및 설치를 요청한다. 요청된 확장 프로그램이 내부 저장소에 존재하면 해당 버전을 제공하고, 존재하지 않거나 새로운 버전인 경우 외부 스토어에서 수집한 뒤 검증 단계로 넘긴다.

2) 검증

유입된 확장 프로그램은 즉시 배포하지 않고 검증 단계를 거친다. 특히 평판이 없거나 신뢰도가 낮은 개발자·기업의 확장 프로그램, 악성 징후가 의심되는 신규·업데이트 버전은 대기열에 보관하여, 일정 기간 동안 형성되는 외부 평판과 추가 분석 결과를 반영할 수 있도록 한다. 이후 정적 분석, 평판 조회, 시나리오 기반 비교, 동적 분석을 순차적으로 수행한다. 정적 분석과 평판 조회에서 수집한 메타데이터는 행위 조합을 표현한 JSON 구조로 변환되어 Vector DB에 저장된 악성 행위 시나리오와 비교되며, 유사도가 높은 시나리오가 발견되면, 해당 시나리오를 기준으로 동적 분석을 수행해 실제 실행 환경에서의 위험 징후를 점검한다[3]. 이러한 검증 결과는 확장 프로그램별 위험 징후와 취

약점을 한눈에 확인할 수 있도록 (표 1)과 같이 요약된다.

(표 1). RAG 아키텍처를 통한 악성 시나리오 매칭 결과 예시

Extension ID	악성 사례 및 근거	매칭 시나리오
obifanppcpchlehkjipahhphbcbjekfa	텔레그램 웹 계정의 세션 탈취 및 계정 가로채기	risk=HIGH scenario=session_storage_exfiltration_reference
jcbiifklmgnkppebelchllpdbnibihel	사용자가 방문하는 모든 웹사이트의 스크린샷을 찍어 원격 서버로 전송	risk=CRITICAL scenario=page_screenshot_or_content_capture
fpeabamapgecnidibdmjoepaiehokgda	AI 기능을 제공하는 척하면서 30개 이상의 원격 명령 실행	risk=MEDIUM scenario=browser_automation_remote_control
eebihieclccoidddmjcencomodomdoei	사용자가 작성하는 내용을 읽고, 민감 데이터를 외부 서버로 탈취	risk=MEDIUM scenario=input_change_event_collection
iefpkdilnfhogjbkhgnliaomoldgkdlj	AI 서비스에 사용하는 API 키를 탈취하여 원격 서버로 전송	risk=MEDIUM scenario=input_change_event_collection

3) 저장·재평가

분석 결과는 안전, 검토 필요, 위험으로 분류된다. 안전에 해당하는 확장 프로그램은 내부 저장소에 저장되어 사용자에게 제공되며, 검토 필요 또는 위험에 해당하는 경우에는 보안 담당자에게 알림으로 전달되어 최종적으로 승인 또는 폐기 여부를 결정한다. 그러나 저장 이후에도 검증은 종료되지 않으며, 이미 승인된 확장 프로그램이라도 새로운 버전이 배포되거나 평판이 변경되면 Retro 분석을 통해 버전·평판 변화를 주기적으로 점검하고, 새로운 징후가 발견되면 재평가가 이루어진다.

3. 결론

본 연구는 브라우저 확장 프로그램을 단순한 편의 기능이 아니라, 자동 업데이트와 높은 권한 구조를 가진 공급망 구성 요소로 보고, 유입·검증·저장·재평가로 이루어진 순환 구조의 다층 검증 체계를 제안하였다.

1) 공급망 보안 가능성

공급망 관점에서 볼 때, 브라우저 확장 프로그램은 단일 시점의 안전성만으로는 충분하지 않으며, 이후 버전·권한·외부 통신·평판 변화를 계속 추적해야 한다. 앞서 제안한 구조는 대기열 운영, 버전 추적, 각종 분석을 통합함으로써 확장 프로그램의 전 수명 주기에서 위험 징후를 반복적으로 재평가할 수 있게 하며, 이를 통해 기존의 설치 시점 중심 접근보다 공급망 보안 요구에 더 부합하는 운영 모델을 제공한다.

2) 타 Extension 확장성

특히 Chrome과 같은 특정 브라우저나 Nexus와 같은 개별 Repository Manager에 종속되지 않고 구현한 것이 특징이다. 이러한 설계는 “외부 저장소를 통해 배포되고, 자동 또는 주기적으로 업데이트되며, 버전·권한·평판 변화에 따라 위험도가 달라지는 Extension“이라는 공통점에 기반하므로, Chrome뿐 아니라 Edge·Firefox, VS Code Extension, 각종 플러그인 및 데스크톱 애플리케이션에도 동일한 방식을 적용할 수 있다. 따라서 자산들을 외부 생태계의 신뢰가 아닌 내부 보안 거버넌스 하에 관리·통제할 수 있는 기반을 마련하였다.

참고문헌

[1] 김대원, 강동욱, 최용제, 이상수, 최병철, “공급망 보안기술 동향,” 전자통신동향분석, 제35권, 제4호, 2020, pp. 149~157.

[2] Shahriar, H., Weldemariam, K., Zulkernine, M., and Lutellier, T., “Effective Detection of Vulnerable and Malicious Browser Extensions,” Computers & Security, Vol. 47, 2014, pp. 66~84.

[3] Hammi, B., Zeadally, S., and Nebhen, J., “Security Threats, Countermeasures, and Challenges of Digital Supply Chains,” ACM Computing Surveys, Vol. 55, No. 14S, Article 316, 2023, pp. 316:1~316:40.

[4] Picazo-Sanchez, P., Ortiz-Martin, L., Schneider, G., and Sabelfeld, A., “Are Chrome Extensions Compliant with the Spirit of Least Privilege?,” International Journal of Information Security, Vol. 21, No. 6, 2022, pp. 1283~1297.

[5] 정준영, 구도훈, 김종민, 김주원, 이주명, 이상현, 이강석, 최지헌, “브라우저 확장 프로그램을 활용한 웹 취약점 분석 방안,” 차세대 보안리더 양성 프로그램(BoB) 10기 논문집, 2022.