오픈소스 SW 공급망 공격 대응기술 동향

이희조*

* 고려대학교 컴퓨터학과 교수, heejo@korea.ac.kr

요 약

오픈소스를 활용하는 것은 비단 개발 시간을 단축하는 것만이 아니라, 혁신 기술을 채용함으로써 경쟁력 있는 서비스를 개발하는데 매우 중요한 요소가 되었다. 하지만, 오픈소스는 잘 사용하지 않으면 라이선스 위반으로 인한 법적 위협뿐 아니라 보안취약점으로 인한 공격, 그리고 다양한 공급망 공격으로 인해 되돌릴 수 없는 치명적인 결과가 초래될 수 있다. 본 고에서는 최근에 급증하고 있는 오픈소스 공급망 공격 동향을 살피고, 공급망 보안 강화를 위한 오픈소스 신뢰도 판단 방법, 개발 및 유통과정에 소프트웨어 부품명세서인SBoM활용, CVE와 같이 치명적 취약점 패치여부 확인 등 오픈소스를 안전하게 활용하기 위한 대응 방안을 소개한다.

01 오픈소스 SW 공급망 공격 대응기술 동향

I 서 론

"내가 더 멀리 보았다면 이는 거인들의 어깨 위에 서 있었기 때문이다."

- 아이작 뉴턴

오픈소스를 활용하는 것을 종종 거인의 어깨에 서는 것으로 비유되고 있듯이, 경쟁력 있는 서비스를 개발하는데 오픈소스를 이용하는 것은 자연스러운 현상이 되었다. 이는 비단 개발시간을 단축시켜 경쟁업체보다 빨리 서비스를 개발하는 것뿐 아니라 혁신기술을 채용할 수 있게 하고, 더 높은 품질의 코드를 개발하게 하여 기술 경쟁력을 높이는데도 중요한 역할을 하기 때문이다.

하지만, 오픈소스를 활용하는데 있어서 주의를 기울이지 않으면 여러 문제가 발생할 수 있으며, 이러한 원인으로는 아래의 인식 오류로부터 기인하고 있다.

"오픈소스는 공짜 소프트웨어이다."

"최신 버전 오픈소스는 보안취약점 패치가 되어 있어 안전하다."

"많은 사람들이 사용하는 오픈소스가 더 보안 수준이 높다."

오픈소스는 공짜가 아니고 라이선스를 준수하여야 한다. 또한, 최신 버전이라도 또 다른 오픈소스를 가져다 사용하는 경우 서브 컴포넌트는 업데이트 되지 않는 경우가 종종 발생하고 있어 서브 컴포넌트 업데이트 주기가 신뢰도 평가에 중요한 요소가 된다. 또한, 많이 사용하는 유명 오픈소스들에서도 치명적인 취약점들이 계속 발견되고 있어서 보안 취약점 동향을 파악하여 적극적인 보안관리를 하지 않으면 큰 피해를 입을 수 있다.

본 고에서는 오픈소스를 선택하고 사용하는데 있어서 법적, 기술적 위험관리의 필요성과 대응기술의 동향을 파악하여 효과적이며 안전하게 오픈소스를 활용하기 위한 방안에 대해 논의한다.

Ⅱ 오픈소스 SW 공급망 공격 동향

1. 오픈소스 활용 현황

오픈소스의 활용은 IT산업 외에도 여러 산업분야에 AI, IoT, 빅데이터 적용이 확대되면서 급격하게 증가하고 있으며, 이는 다양한 통계에서도 이를 반영하고 있다. 오픈소스 저장소 깃허브는 2021년 10월 현재 2억개 이상의 프로젝트의 저장소로 사용이 되고 있고, 6천5백만명 이상의 개발자가 활동하고 있으며, 3백만개 이상의 회사와 조직이 이용하고 있다. 또한, 2021년 현재 Java, JavaScript, Python, .NET 4개 언어로 된 오픈소스 배포에 이용되는 패키지 매니저 Maven, npm, PyPi, NuGet에 등록된 오픈소스만 해도 1년전에 비해 20% 증가한 3천7백만 여개의 소프트웨어가 등록되어 있고, 다운로드 횟수는 전년도에 비해 73%나 증가하였다[1]. 또한, 상용 포함 소프트웨어의 98%가 오픈소스를 활용하고, 코드의 75% 이상이 오픈소스로 구성이 되어있다[2]. 또한, 소프트웨어의 84%에서 하나 이상의 보안 취약점이 발견되었으며, 65%의 경우는 라이선스 충돌이 발견되었다.

그림에서와 같이 깃허브에서 C/C++ 프로젝트 중에 포크횟수 기반으로 상위 10개의 프로젝트를 보면, 인공지능에 활용되는 구글 Tensorflow, 리눅스 커널, 비트코인 등은 원본 소스가 복사되어 이미 각각이 수만 개의 새로운 프로젝트에서 새로운 이름으로 개발이 진행되고 있음을 알 수 있다.

그림 1 깃허브 C/C++ 상위 10개 프로젝트의 포크횟수, CSSA, 2021년 10월

Rank	Project Name	Forks	Area
1	Tensorflow	86 K	AI
2	OpenCV	48 K	AI
3	Linux	39 K	IoT
4	Bitcoin	30 K	Blockchain
5	Git	22 K	OSS
6	Redis	20 K	DB
7	Darknet	18 K	DB
8	Pytorch	14 K	AI
9	Electron	13 K	Web
10	Arduino	12 K	IoT

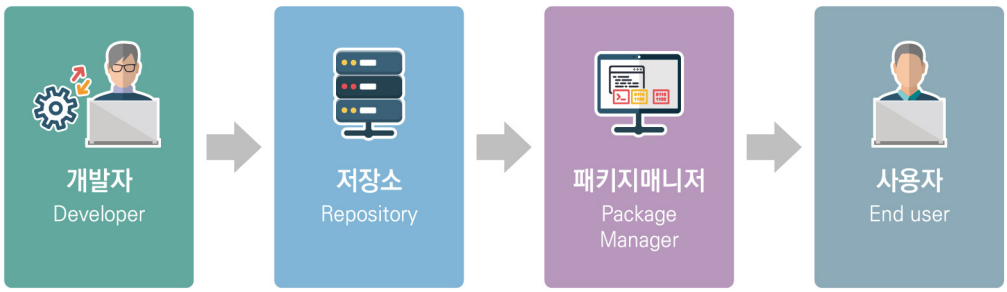
2. 공급망 공격 유형

소프트웨어 공급망(Supply Chain)은 소프트웨어가 개발자에 의해 개발된 후 사용자에게 도달하기까지 여러 단계로 구성된다. 개발자의 개발 환경부터 소스 코드 저장소, 최종 소프트웨어가 배포되는 플랫폼까지 포함될 수 있으며, 이러한 공급망은 각 소프트웨어를 개발하는 오픈소스 커뮤니티와 개발사, 그리고 소프트웨어 특성에 따라 달라질 수 있다. 특히, 오픈소스의 경우 재사용 빈도가 높아 사유 소프트웨어에 비해 공급망이 더 길어질 수 있으며, 상호 재사용으로 인해 복잡한 의존성을 가지고 있는 경우 또한 빈번하게 발생하고 있다.

이를 일반화하여 표현하면 오픈소스 공급망은 [그림 2]와 같이 개발자, 저장소,

패키지 매니저, 사용자 네 단계로 구성되고 있다. IoT 기기를 이용한 금융 서비스의 경우는 공급망이 더욱 길어질 수 있어서, [그림 2]의 사용자 부분이 제조사, 서비스 제공자, 사용자로 6단계로 구성 된다. 공급망의 각 단계는 공격 목표지점이 될 수 있으며, 모든 단계의 보호가 되지 않으면 공격이 가능해진다. 이는, 생태계의 상호 의존성을 보여주는 것으로, 금융기관은 솔루션이나 기기를 공급받는 제조사뿐 아니라 오픈소스 생태계의 보안관리 체계에도 관심을 기울여야 하는 이유가 되고 있다.

그림 2 오픈소스 공급망의 4단계 구성



2015년 이후 2021년 현재까지의 공급망 공격에 사용된 기법을 분류하여 보면, 공급망 단계별로 상이한 공격 형태 및 다양한 공격기법이 존재함을 알 수 있다. 오픈소스 공급망의 주요 단계에 관련된 대상 시스템으로는 개발서버, 저장소, 패키지 관리자가 있으며, 단계별 공격 유형은 아래와 같이 9개 유형을 포함하게 된다.

2021년 공급망 공격이 급증하고 있어서, 전년도에 비하여 650% 증가하였다[1]. 공급망 공격의 예로는 2020년 말 보고된 솔라윈즈 공격 외에도 2021년 7월 발생한 카세야 랜섬웨어 공격도 공급망 공격의 사례라 볼 수 있다. 최근 통계로 보면 3.1 의존성 혼돈공격과 3.2 타이포스쿼팅 공격과 같이 네임스페이스 공격 유형이 가장 자주 발생하고 있다.

그림 3 공급망 주요 단계의 해당 시스템에 대한 공격 유형

공급망 단계		공격 유형		공격 사례
1	개발 서버	1.1	서버 침투 및 악성코드 삽입	SolarWinds(2020-2021), Codecov(2021), Kaseya(2021)
		1.2	프로젝트 파일 대상 악성코드 배포	Octopus Scanner(2020), rest-client(2019)
		1.3	악성 IDE 배포	Xcode Repackaging(2015)
2	저장소	2.1	오펀 패키지 탈취 (Orphan Package Takeover)	Arch User Repository acroread(2018)
		2.2	계정 탈취	Sawfish phishing campaign(2020)
3	패키지 매니저	3.1	의존성 혼동 (Dependency Confusion)	Alex Birsan's Research(2020-2021)
		3.2	타이포스쿼팅 공격 (Typo-squatting Attack)	Electron(2020), atlas-client(2020)
		3.3	악성 패키지 배포	1337qq-js(2020)
		3.4	계정 탈취	RubyGems-strong_password(2019)

의존성 혼돈(dependency confusion) 공격이란 정상 소프트웨어의 이름과 동일한 이름의 새로운 버전 소프트웨어를 공급망에서 배포하도록 하는 공격으로, 자동화된 빌드 도구들이 다운로드 하도록 하거나 자동 업데이트로 인해 악성코드 전파 속도가 높아지는 경우이다. Alex Birsan은 2021년 2월 기업이 개별적으로 사용하는 패키지 이름과 동일한 이름을 공개 패키지 관리자 사이트에 등록하는 방식으로 페이팔(Paypal), 애플(Apple), 쇼피파이(Shopify), 넷플릭스(Netflix), 우버(Uber), 옐프(Yelp) 등 35곳이 넘는 기업을 손쉽게 공격이 가능함을 보였다[10].

타이포스쿼팅(typo-squatting) 공격은 정상 패키지와 유사한 이름의 패키지를 등록하여 사용자로 하여금 악성 패키지를 다운받아 사용하게 하는 경우를 말한다. 2020년 RubyGems 패키지 매니저에 700개의 악성 타이포스쿼팅 패키지가 발견이 되었으며, 예를 들면 acpc_poker_types라는 이름의 정상 패키지를 노린 acpc-poker_types라는 이름의 악성 패키지를 등록한 사례가 이에 해당한다.

공급망 공격은 한번의 공격으로 다수의 소프트웨어에 영향을 끼치고, 더 나아가 공격 대상 패키지에 의존하는 소프트웨어와 이를 사용하는 사용자까지 공격하는 증폭 효과를 가지고 있어, 치명적인 결과를 초래할 수 있게 된다. 또한, 오픈소스는 특성상 전세계 개발자 누구라도 참여하여 개발과 공유를 장려하고 있으므로,

악의적인 사용자 또한 공급망에 쉽게 접근이 가능하기 때문에, 공격은 쉽고 효과는 높아 공급망 공격은 점점 증가하는 추세에 있다. 이에 따라 소프트웨어의 공급망 보안의 중요성이 높아지고 있으며, 공급망 보안관리를 제도화하는 움직임이 국가별, 산업별로 증가되고 있다.

3. 오픈소스 보안 취약점

"보는 눈이 많다면 어떤 버그도 쉽게 잡을 수 있다." - 리누스 법칙

오픈소스는 많은 사람들이 공개적으로 사용하고 이에 따라 보안 문제가 발생하면 바로 발견하여 해결할 수 있으므로 보안성이 높다는 주장이 리눅스 개발자의 이름을 따서 "리누스 법칙"이라 한다. 하지만, 모든 오픈소스가 이에 해당하지는 않아서 오픈소스라는 것만으로 안전이 담보되지는 못한다. 따라서, 오픈소스를 사용할 때는 오픈소스의 선정 및 평가, 그리고 사용하는 모든 라이프사이클에서 새로운 취약점에 노출되지는 않는지 정기적 보안 관리가 필요하게 된다.

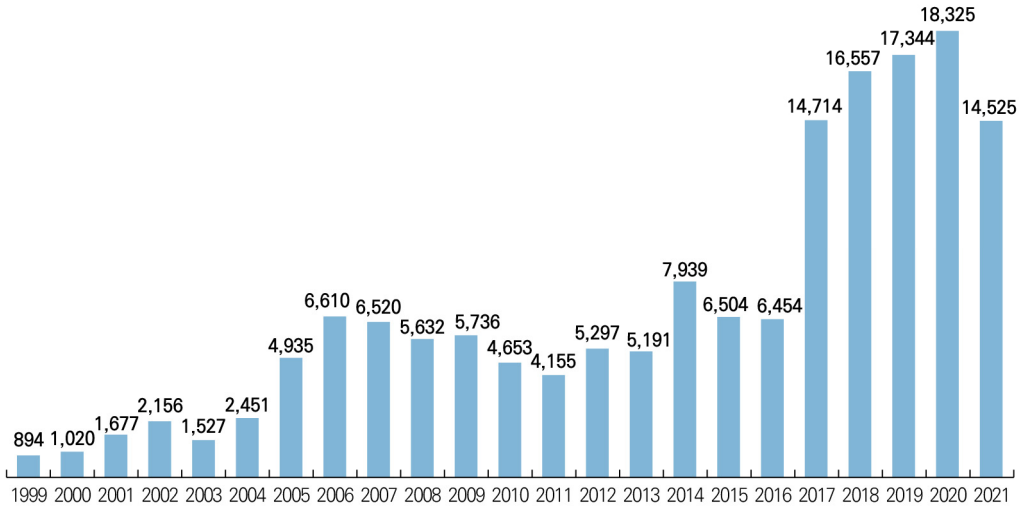
국제표준 취약점 식별번호로 사용되는 CVE(Common Vulnerabilities and Exposures)의 경우 2021년 9월 현재 16만개의 취약점을 공개하고 있으며, 하루 평균 50개의 신규 취약점이 발견되고 있다.[1] 연도별 취약점 추이는 아래 [그림 4]에서 보여지고 있으며, 최근 5년간 신규 취약점이 그 이전에 비해 2배 이상 증가하였고, 매년 더 증가하는 추세를 보이고 있다.

CVE 취약점은 금융서비스의 공격에도 이용이 되고 있어서, 2014년 HeartBleed 취약점을 이용해 SSL/TLS로 보호되는 웹 또는 VPN 서비스에 대한 공격 등 많은 사례가 포함된다. 또한 최근에는 다양한 기기에서 사용중인 4개의 오픈소스 TCP/IP 스택에서 Namewreck 취약점이 발견되어 전세계 적으로 1억대 가량 기기가 공격에 노출이 되었고, 2021년 4월에 마이크로소프트에서 발견한 BadAlloc 취약점은 임베디드 기기와 산업용 OT기기에서도 많이 사용하는 다수의

1) https://cve.mitre.org

RTOS 에서 25개의 취약점이 발견된 경우이다. 이처럼 매일 발견되어 공개되는 취약점들은 금융서비스에 위협이 되는 사항으로 서비스에 영향을 끼치는지 동향 파악 및 관리가 필요하다.

그림 4 새롭게 공개되고 있는 CVE 취약점 연도별 트랜드

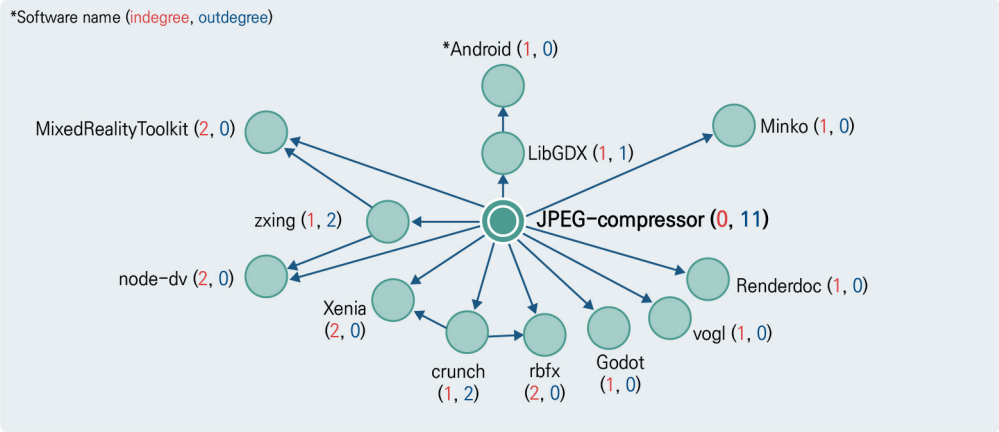


고려대 소프트웨어보안연구소(CSSA)[2]의 최근 연구에 의하면, 많은 보안 기술 및 솔루션에서 사용하고 있는 CVE 정보의 2%는 취약한 소프트웨어에 대한 정보가 잘못되어 보안 패치 등 대처가 늦어지고 있음이 발견되었다[11]. 의학계에서 전염병의 첫 번째 감염자를 탐지하는 것은 광범위한 확산을 방지하기 위해 매우 중요한 요인이 된다. 마찬가지로, 소프트웨어 보안취약점의 최초 근원지를 정확하게 탐지하는 것이 전파된 취약점 조기 탐지에 큰 영향을 미친다는 점에 착안하여 자동화된 방식으로 정확하게 취약점의 최초 근원지, 즉 취약점이 최초로 발생한 지점을 의미하는 'Vulnerability Zero'를 탐지하는 기술을 개발했다. 취약점의 올바른 최초 근원지가 제공되는 경우, 해당 CVE취약점을 평균 6개월 이내에 패치가 되고 있었던 반면, 근원지가 잘못된 CVE의 경우 해당 취약점이 있는 오픈소스를

2) https://cssa.korea.ac.kr

사용 중인 소프트웨어는 2년이 넘어도 패치 되지 않고 지연되고 있음을 확인할 수 있었다[4].

그림 5 JPEG 취약점이 Android로 보고된 CVE의 타 오픈소스로의 취약점 전파경로



[그림 5]에서는 JPEG 취약점이 Android 취약점으로 등록이 되어있는 CVE-2017-0700 취약점이 다른 오픈소스에도 전파된 상황을 보여준다. 즉, JPEG-compressor를 사용하는 LibGDX, Godot 등은 취약한지를 알 수 없고 패치도 지연되고 있는 반면, 구글이 관리하는 Android만 빨리 패치가 가능했던 경우이다. 이는 구글처럼 많은 예산과 자원을 들여 적극적으로 취약점 관리를 하고 버그바운티도 운영하는 경우 더 많은 취약점 정보가 흘러가고, 반면 취약점 관리를 하지 않는 쪽은 취약점이 발견된지도 모르고 있어 해커의 공격으로 더 큰 피해를 입게 됨을 보여준다.

따라서 제품이나 서비스에서 발견되는 취약점 정보를 받아들일 수 있는 취약점 관리 체계를 운영하는 것이 취약점 정보를 먼저 입수하여 대처 함으로서 고객의 피해를 줄일 수 있게 해준다는 것을 알 수 있다. 취약점 관리체계는 최근 국가별 법제도 및 산업별 표준으로 요구사항이 증가되고 있으며, 2021년 5월 미국의 사이버 보안 행정명령의 공급망 보안, ISO/IEC 29147:2018 취약점 신고 및 책임 공개

국제표준, UNECE WP.29 자동차보안 취약점관리체계 등이 이에 해당한다.

이러한 보안관리 체계는 표준 및 인증의 요구사항을 위한 것뿐만 아니라 담당 팀과 프로세스를 적립해 둠으로써 보안 문제에 대한 빠른 대처에도 큰 기여를 할 수 있게 한다. 2021년 7월 발생한 카세야 랜섬웨어 공격은 사건발생 3개월 전인 4월에 네덜란드 팀에서 신고한 7개 CVE 취약점 중 패치 개발이 지연되었던 하나의 CVE가 공격에 이용되어 발생한 사건이다. 결과적으로 고객의 고객사 1500여개 기관에 랜섬웨어가 배포되었고 800억의 몸값이 요구되었다. 이미 보고되었던 취약점을 패치 개발과 배포 등 대응이 지연되어 발생한 사건으로 취약점 신고 및 관리 체계의 중요성을 보여주고 있다. 보안취약점 관리를 위한 조직과 절차에 대해서는 국제표준인 ISO 29147을 참고하여 구축이 가능하다[15].

4. 오픈소스 라이선스 위반

오픈소스는 공짜 소프트웨어가 아니고, 라이선스에 정의된 규칙을 따라 사용하여야 한다. 라이선스의 종류는 OSI(Open Source Initiative)에 의해 검증하여 SPDX 표현으로 공개한 것만 114개에 이르고 있으며, 보안취약점 자동분석 도구 중 하나인 아이오티큐브의 래브라도는 501개의 라이선스 유형을 관리하고 있다. 이중에 MIT, BSD, Apache등 보다 자유로운 라이선스도 있지만, GPL(GNU General Public License)처럼 카피레프트 철학이 반영이 되어있는 라이선스도 존재한다. 대표적인 사례인 리눅스 커널은 GPLv2를 따르고 있고, 이를 이용하는 Android, Tizen, WebOS등의 운영체제와 이를 기반으로 개발된 스마트기기 또한 변경한 소스코드를 오픈소스로 공개해야 하는 의무가 부과된다.

오픈소스 라이선스 위반으로 인한 소송이 구글, 마이크로소프트, 시스코, IBM 등의 글로벌 기업뿐 아니라 한글과컴퓨터 등 국내 기업을 대상으로도 증가하고 있다[8]. 국내외 기업에서는 법무팀의 대응이 선제적으로 이뤄지고 있으나, 오픈소스는 법적인 이슈 외에도 보안 이슈가 점차 증가하고 있어 법무팀과 보안팀이 함께 효과적인 관리체계를 구축해 나갈 필요가 있다.

공개 취약점 정보의 불완전성과 매일같이 찾아지는 취약점 정보 홍수 속에서도 안전한 공급망 관리와 서비스 제공을 위해서는 오픈소스의 목록관리 및 보안성 검증, 업데이트 필요시 즉각적인 실행 체계 구축 등을 위한 오픈소스 거버넌스 체계의 구축이 시급하다.

Ⅲ 공급망 보안 강화방안

1. 오픈소스 신뢰도 판단 기준

어떤 오픈소스를 사용할 것인가를 판단하는 기준은 여러 가지가 제시되고 있다. 앞서 소개한 바와 같이 포크 횟수나 스타 개수 또는 참여 개발자의 수처럼 인기 순위(popularity) 기반의 척도, 또한 얼마나 자주 업데이트를 하고 새로운 버전을 배포하는지 활동중심의 척도(activity), 그리고 버전관리와 문서작성 등의 성숙도(maturity) 기반의 척도 등 다양한 방식이 제안되고 있다[1]. 하지만, 오픈소스의 보안 측면에서 가장 중요하게 보아야 할 것은 “업데이트 빈도”를 빼놓을 수 없다.

다음은 고려대 CSSA에서 깃허브의 스타 갯수 기준으로 최상위 오픈소스 2000개를 아이오티큐브[3] 취약점 자동분석 플랫폼을 이용하여 분석했을때, 유명 오픈소스의 최신버전임에도 불구하고 15%에서는 알려진 CVE 취약점의 패치가 이뤄지지 않은 것을 확인할 수 있었다[5].

이는 오픈소스는 또 다른 오픈소스를 이용하여 개발을 하게 되는데, 내부에 있는 서브 컴포넌트가 업데이트 되지 않아서 발생하고 있다. 즉, 오픈소스가 최신 버전이라고 하여도 내부에 사용 중인 또 다른 오픈소스 컴포넌트는 최신 버전이 아닌 경우로, 적게는 3개월전 버전에서부터 길게는 10년전 버전의 컴포넌트가 사용 중임을 알 수 있었다[5,12].

3) IoTcube 퍼블릭 플랫폼은 https://iotcube.net 을 통해 전세계 누구라도 무료로 사용이 가능하다.

이와 같이 오픈소스의 최신버전이라도 서브 컴포넌트의 업데이트가 되지 않음으로써 보안 취약점에 노출될 수 있고, 이에 따라 최상위 소프트웨어뿐만 아니라 서브 컴포넌트에서 사용 중인 소프트웨어도 관리를 해야 하는 것을 나타내고 있다.

그림 6 C와 C++ 언어로 구성된 최신버전 오픈소스에서 CVE 취약점이 발견된 오픈소스 목록, CSSA 2020

Name	# CVE	Area
Emscripten	15	Compiler
Turicreate	14	AI
Godot	10	Game
Mongo	2	Database
ArangoDB	2	Database
OpenCV	1	Vision

Name	# CVE	Area
FFmpeg	15	Media
kbengine	14	Game
Linux	13	OS
RaspberryPi	13	OS
Freebsd	7	OS
OpenSSL	2	Security

이에 따라, 오픈소스의 신뢰도 평가 척도는 서브컴포넌트를 포함한 업데이트 주기를 중심으로 해야 한다는 주장과 일치하는 결과를 보이고 있다.

2. SBoM 부품명세서 관리

소프트웨어 공급망 보안의 첫걸음은 사용중인 소프트웨어의 구성요소를 파악하는 것에서 시작할 수 있다[6]. 즉, 해당 제품이 어떤 소프트웨어의 어떤 버전들로 구성이 된 것인지를 나타내며(a “list of ingredients” in software) 이 정보를 소프트웨어 명세서, 혹은 SBoM(Software Bill of Materials)이라고 쓰고 “에스봄”이라 읽는다. IoT 기기와 금융을 포함하는 스마트 서비스, 그리고 다양한 기기가 복잡하게 운영되는 스마트 공장 및 자동차 등에서 SBoM을 파악하고 있으면,

라이선스 위반여부 및 새로 발견되어 보고된 취약점에 해당이 되는지 즉시 파악하여 대처함으로써 피해를 최소화할 수 있게 된다[9].

그림 7 SBoM을 활용한 소프트웨어 공급망 보안 강화

이에 따라, 미국에서는 2021년 5월 바이든 대통령의 행정명령으로 공급망 보안을 위한 SBoM 제공 의무화를 추진하고 있으며, 헬스케어 등 IoT 서비스에 시범 적용한 경험을 바탕으로 사이버보안 수준의 획기적 향상을 기대하고 있다.

제품에 포함된 상용 및 오픈소스 소프트웨어 컴포넌트 정보 등 소프트웨어 재료 명세서로서 SPDX, CycloneDX, SWID 등이 대표적인 표준이다. 소프트웨어 구성 요소 및 이들 간의 공급망 관계를 나열하고 식별하는 소프트웨어 재료 명세서로서 이름, 버전정보 외에 공급망 정보를 포함하고 있어서, 라이선스 등의 컴플라이언스 준수 여부뿐 아니라, 취약점 발견시 해당 소프트웨어에 취약점 존재 여부를 빠르게 파악하여 패치 시간을 급격히 단축하게 하는 등 보안 관리를 용이하게 한다. 특히, 부품업체가 상위 제품 생산자에게 납품할 때뿐 아니라 제품을 이용한 서비스를 고객이 도입할 때, 그리고 정기적으로도 새롭게 발견된 취약점을 손쉽게 확인이 가능함으로써 보안 문제점의 파악 및 해결을 도와주게 한다.

그림 8 자체 코드와 오픈소스, 외부 라이브러리를 부분별로 시각화한 소프트웨어 맵과 SBoM 제공사례

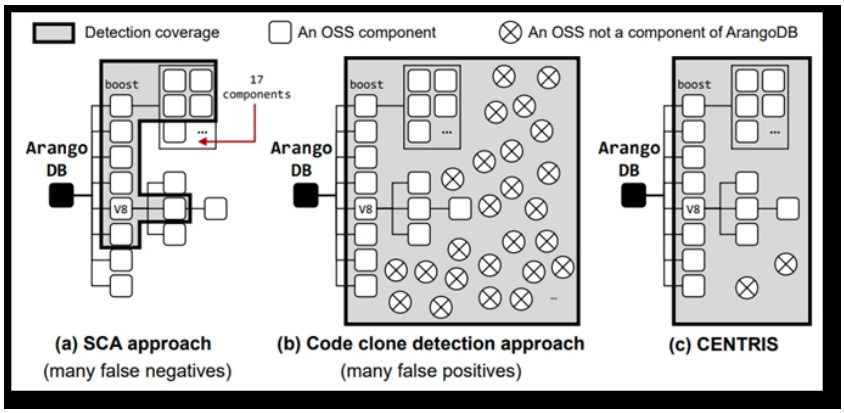
3. 오픈소스 컴포넌트 식별

오픈소스 소프트웨어는 재사용 되는 과정에서 원본 구조 및 코드가 수정되는 경우가 잦다. 고려대 CSSA에서 수행한 연구에 의하면 자주 사용되는 유명한 오픈소스 소프트웨어들의 95%는 코드 혹은 구조 수정과 함께 재사용되고 있다[12]. 기존의 알고리즘은 5%의 수정 안된 오픈소스는 정확히 탐지하지만, 95%에 이르는 수정된 오픈소스가 포함된 소프트웨어에서는 오탐이 많아 탐지 정확도가 10%에 그친다는 것을 알 수 있었다. 반면 코드 분할 및 중복 제거 알고리즘을 기반으로 동작하는 Centris는 수정된 오픈소스 구성요소가 포함되어 있어도 90% 이상의 정확도로 빠르고 정확하게 식별해 낼 수 있다[3].

[그림 8]은 오픈소스인 ArangoDB에 존재하는 17개의 서브 컴포넌트에 대해 기존 방식들과 Centris의 탐지 결과를 비교하여 보여주고 있다. 기존 SCA방식으로는 정상적인 컴포넌트를 못찾는 미탐이 많이 발생하고, 코드클론 탐지방식으로는 일부 코드만보고 해당 컴포넌트가 있다는 결과를 보고하면서 많은 오탐이 발생하게 된다. 이에 반하여 Centris는 17개의 컴포넌트를 모두 찾아주고 오탐도

현저히 작은 수준임을 보여주고 있다.

그림 9 ArangoDB 분석 결과 타 방식에 비해 월등히 높은 Centris 정확도

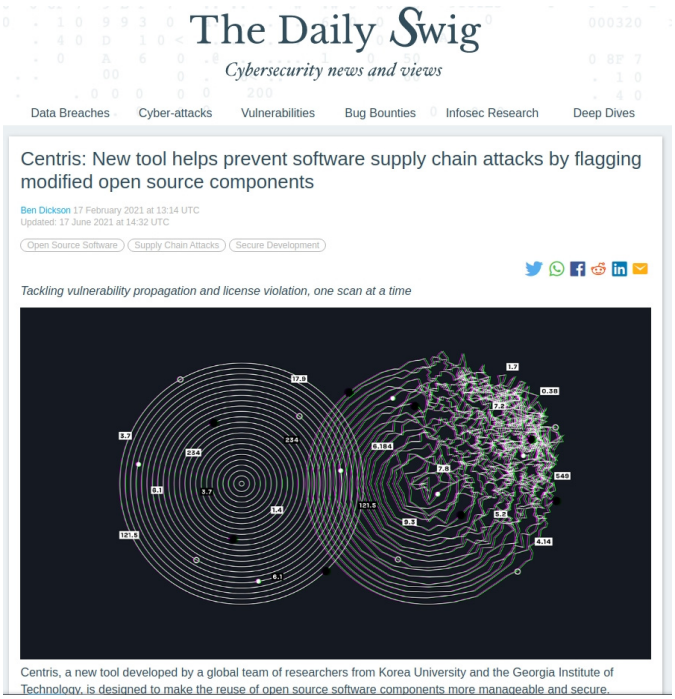


[그림 9]는 오픈소스인 ArangoDB에 존재하는 17개의 서브 컴포넌트에 대해 기존 방식들과 Centris의 탐지 결과를 비교하여 보여주고 있다. 기존 SCA방식으로는 정상적인 컴포넌트를 못찾는 미탐이 많이 발생하고, 코드클론 탐지방식으로는 일부 코드만보고 해당 컴포넌트가 있다는 결과를 보고하면서 많은 오탐이 발생하게 된다. 이에 반하여 Centris는 17개의 컴포넌트를 모두 찾아주고 오탐도 현저히 작은 수준임을 보여주고 있다.

Centris는 소프트웨어 원심분리기(Centrifuse of Software)의 의미를 가지고 있으며, 아이오티큐브 퍼블릭 플랫폼을 이용해 공개가 되었으며,[4] 래브라도[5]에도 적용이 되어 기업에서도 개발 또는 검수단계에 오픈소스 소프트웨어 구성요소를 정확하게 식별하고 SBoM을 관리하는 용도로도 사용할 수 있다. [그림 10]에서는 오픈소스가 재사용 빈도가 높아지면서 취약점의 전파경로로 사용될 수 있으며, Centris를 이용하여 수정하여 재사용하는 경우라도 서브 컴포넌트를 정확히 식별하여 공급망 보안을 높일 수 있다는 내용을 소개하고 있다.

4) https://iotcube.net/centris
5) https://labrador.iotcube.com

그림 10 소프트웨어 공급망 보안에 Centris가 효과적으로 활용될 수 있음을 소개, The Daily Swig 2021년 6월 17일



4. CVE 취약점 관리

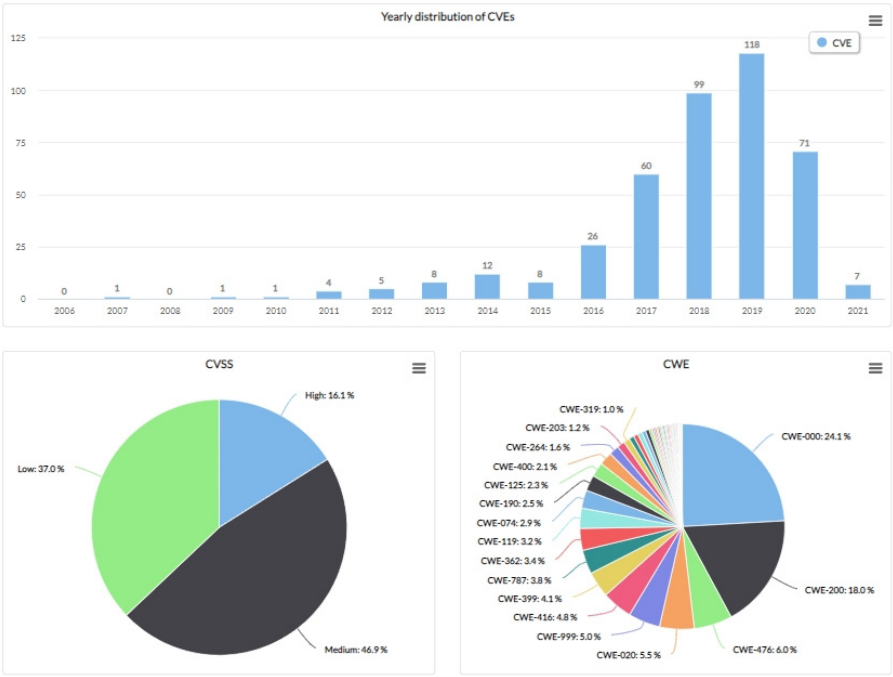
해킹 공격의 90%는 소프트웨어 취약점을 이용한 공격이라는 US-CERT 보고가 있듯이, 치명적인 취약점을 패치 업데이트하는 것은 보안의 매우 중요한 요소이다. 하지만, 소프트웨어의 취약점이 존재하는지 여부는 공급망 최상단에 있는 소프트웨어 또는 제품 명으로는 구분할 수 없다. 예를 들어, 삼성 갤럭시 스마트폰의 취약점은 삼성 갤럭시로 보고된 CVE 뿐만 아니라 안드로이드와 크로미움 브라우저 등 스마트폰에서 사용되는 SBoM 컴포넌트들 각각에 해당하는 CVE 취약점이 있는지를 확인하여야 한다.

아이오티큐브의 퍼블릭 버전을 이용하여 최신 스마트워치 펌웨어를 분석하였을 때, 3천8백만 라인으로 이뤄져 있는 코드에서 패치 되지 않은 CVE 취약점이 421개가 존재함을 알 수 있다. 이는 5년전에 릴리즈 된 Linux Kernel 4.9 버전을

사용하고 있는 부분에서 상당수 발견되는 것을 볼 수 있다.

패치가 되지 않은 코드가 반드시 보안 취약점으로 공격에 이용이 된다고 할 수는 없으나, PoC 익스플로잇까지 검증된 CVE이므로 익스플로잇의 가능성을 매우 높다. 따라서, 발견된 취약점 중에 심각도가 높은 취약점의 경우 대처가 필요하며, 버전 업그레이드뿐 아니라 백포팅(backporting) 기술을 적용함으로써 상위 호환성 문제를 없이 검증된 패치코드를 적용할 수 있다. 백포팅은 새로운 버전의 소프트웨어 또는 보안 패치를 낮은 버전에 포팅하는 것을 의미하며, 버전 업그레이드가 어려운 펌웨어나 오픈소스에도 적용이 가능한 해법이 된다.

그림 11 최신 스마트워치에서 오래된 오픈소스 컴포넌트 사용으로 인해 발견된 CVE 취약점 현황



IV 결 론

오픈소스의 활용은 경쟁력을 높여주지만 관리가 필요하다. 여기에는 공급망 보안을 위한 SBoM 리스트 관리, 라이선스 및 보안취약점 관리를 포함하게 된다. 안전하게 오픈소스를 사용하고 법적, 기술적 위험 관리를 위해서는 조직 및 프로세스를 미리 구축해 두어야 한다. 더불어 이와 같은 오픈소스 거버넌스 체계에서 꼭 필요한 것은 빠르고 정확하게 분석하고 새로운 정보를 업데이트 해주는데 도움을 주는 자동화된 오픈소스의 관리도구도 포함될 것이다. 하지만, 오픈소스 관리도구는 아주 강력하고 편리하지만 보조 도구로서 활용하고 최종 판단은 사용자가 하여야 할 것이다. 이를 위해서는 오픈소스 생태계의 특성인 높은 빈도의 재사용, 수정된 오픈소스 컴포넌트의 업데이트 지연, 그리고 소프트웨어 공급망을 통한 공격 등의 문제점을 인식하고 관련 기술과 도구를 활용하여 보안 관리를 해나갈 필요가 있다.

참고문헌

[1] Sonatype, “2021 State of the Software Supply Chain,” Sep. 2021
[2] Synopsys, “Open Source Security and Risk Analysis Report,” 2021
[3] 보안뉴스, “오픈소스 커뮤니티 노리는 공급망 공격, 국내 연구팀 기술로 차단한다,” 2021년 3월 25일
[4] ZDnet, “SW 취약점 최초 근원지 자동 탐색 기술 등장,” 2021년 9월 8일
[5] 이희조, “오픈소스 소프트웨어 보안 취약점과 공급망 보안,” 한국정보보호학회 단기강좌, 2020년 11월
[6] 이희조, “보안취약점 자동분석 플랫폼 IoTcube 소개”, 제5회 IoTcube 컨퍼런스, 2021년 8월
[7] Open source licenses, OSI(Open Source Initiative), 2021 https://opensource.org/licenses/alphabetical

[8] Open source license litigation, 2021, https://en.wikipedia.org/wiki/Open_source_license_litigation

[9] 미국 NTIA, “Software Bill of Materials,” 2021, https://ntia.gov/SBOM

[10] Alex Birsan, Dependency Confusion: How I Hacked Into Apple, Microsoft and Dozens of Other Companies The Story of a Novel Supply Chain Attack, 2021년 2월 10일, https://medium.com/@alex.birsan/

[11] Seunghoon Woo, Dongwook Lee, Sunghan Park, Heejo Lee, Sven Dietrich, “V0Finder: Discovering the Correct Origin of Publicly Reported Software Vulnerabilities”, USENIX Security Symposium, Aug. 2021.

[12] Seunghoon Woo, Sunghan Park, Seulbae Kim, Heejo Lee, Hakjoo Oh, “CENTRIS: A Precise and Scalable Approach for Identifying Modified Open-Source Software Reuse”, Int’l Conf. on Software Engineering(ICSE), May 2021.

[13] Seulbae Kim, Heejo Lee, “Software systems at risk: an empirical study of cloned vulnerabilities in practice”, Computers & Security, Vol. 77, pp. 720-736, Aug. 2018.

[14] Seulbae Kim, Seunghoon Woo, Heejo Lee, Hakjoo Oh, “VUDDY: A Scalable Approach for Vulnerable Code Clone Discovery”, IEEE Symposium on Security and Privacy, May. 22. 2017.

[15] ISO/IEC JTC 1/SC 27, “Vulnerability disclosure”, ISO/IEC 29147:2018