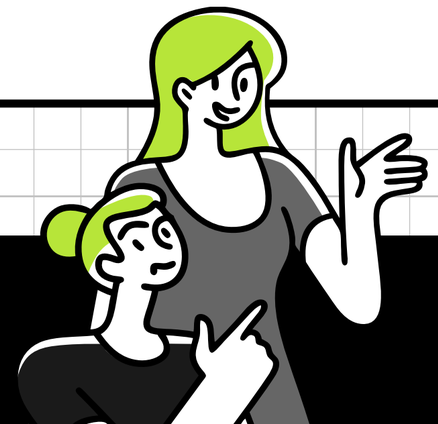
GroMetric

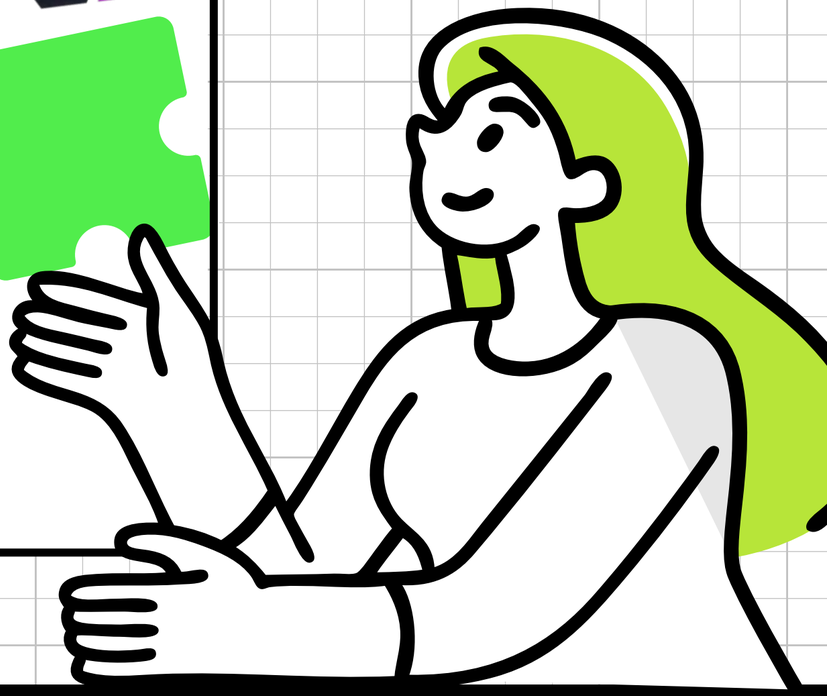
sonatype

3rd
GroMeetUp

Sonatype Nexus repository를 중심으로 본

Hugging Face 오픈소스 AI모델 취약점과 NPM 공격사례





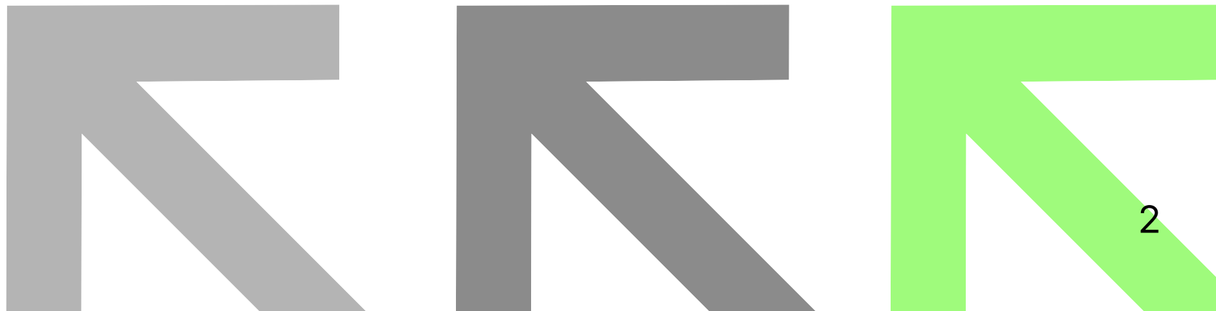
3rd GroMeetUp

Session 01

npm chalk & debug 메인테이너 해킹 및 Shai-Hulud 사태로 본

오픈소스 공급망의 취약성

GroMetric
김관영



목 차

Context

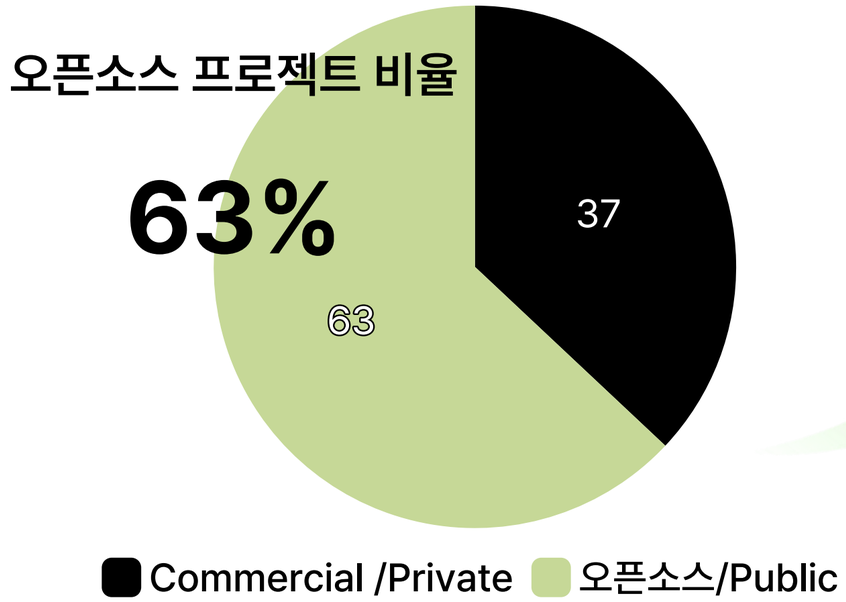
오픈소스 위협의 고도화:

2025 NPM 생태계 회고

Context

오픈소스 위협의 고도화: 2025 NPM 생태계 회고

오픈소스,
Github 저장소에서 차지하는 공급망으로서의 비중이 절대적



실제 기여 활동의 81.5%는
프라이빗 저장소에서 일어나지만,

오픈소스 생태계에 의존

의존성 증가

140% 증가
신규 악성 패키지 발견(직전 분기 대비)

2,889% 폭증
Dropper 유형의 malware 비중(직전 분기 대비)

리소스 고갈
방대한 의존성 트리는 엔지니어링 및 보안 팀이
수동으로 감당할 수 없는 임계점 초과

위협의 증가

2025년 10월 자료 기준 github octoverse

2025년 3분기 자료 기준 sonatype Opensource \|Malware Index

Context

오픈소스 개발자들은 사이버 공격의

'새로운 최전선(New frontline)'

Context

오픈소스 위협의 고도화: 2025 NPM 생태계 회고

[7월]

라자루스 그룹

개발도구 사칭
(npm / PyPI)
위장 패키지

스파이웨어 감염

[8월]

S1ngularity

NX 저장소
취약점을 이용한
CI/CD(Github Action)
오염

비밀키 2천여건 유출

[9월]

npm chalk /debug

메인테이너 계정 탈취
(ATO, Account
Takeover)

**인기 라이브러리 오염
(백도어)**

[11월]

Shai-Hulud2.0

악성 파이프라인 구축

**탐지 우회 및
지속적인 감염**

[12월]

React2Shell

React RSC 원격 코드
실행 취약점
(RCE, Remote Code
Execution)

**서버 무단 원격 제어 허용
CVSS 스코어 10.0**

Context

오픈소스 위협의 고도화: 2025 NPM 생태계 회고

위협의 두가지 얼굴

1 외부의 악성코드 위협 (Malware)

→ 사용자를 속이거나, 악용하거나, 해를 끼치기 위해 의도적으로 제작된 것

- Shai-Hulud2.0
- 라자루스 그룹
- S1ngularity
- npm chalk/debug

2 내부의 치명적 결함 (Vulnerability)

→ 의도치 않은 것으로, 소프트웨어 개발 과정상의 오류로 인해 발생

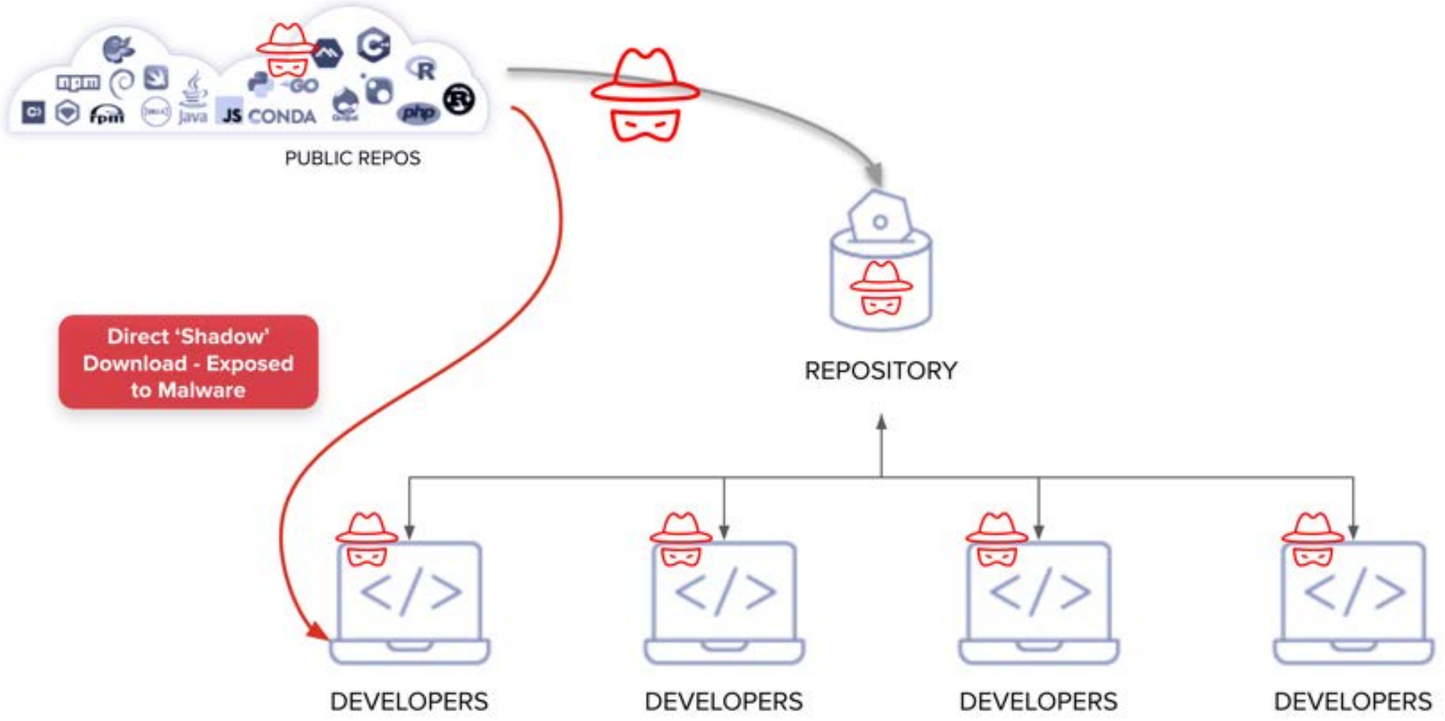
- React2Shell

01. 외부의 악성코드 위협

Malware

DeepDive

오픈소스 Malware 침투 경로와 특징



1) 무방비 상태의 유입 가능성

- 외부망과 내부망 사이의 네트워크 방화벽은 존재하나, 패키지 유입을 검증하는 '컨텐츠 fw' 부재
- 개발자의 npm install 한 번으로 악성 코드가 내부 파이프라인에 영향

2) 공격 목적의 고도화(APT공격과 동일)

- 백도어를 설치하여 장기간 잠복 및 서버 제어권 탈취
- API Key, 자격 증명(Credential), 고객 데이터 탈취 후 판매

DeepDive

01. 외부의 악성코드 위협 (Malware)

공격자들은 성공이 입증된 방법을 더욱 정교하게 다듬고 개선하면서 공격 캠페인을 계속 진행

sonatype-2025-003716 / sonatype-2025-003727로 추적

Case 01
debug/chalk npm 패키지 하이재킹

8개의 인기 npm 패키지(debug, chalk 등을 포함하되 이에 국한되지 않음)를 대상으로 한 광범위한 npm 공급망 공격, 패키지 관리자 계정 탈취(ATO,Account Takeover) 를 통해 수천만의 다운로드 패키지 장악/업데이트에 악성 코드 삽입

sonatype-2025-003810로 추적

Case 02
Shai-Hulud 2.0 캠페인

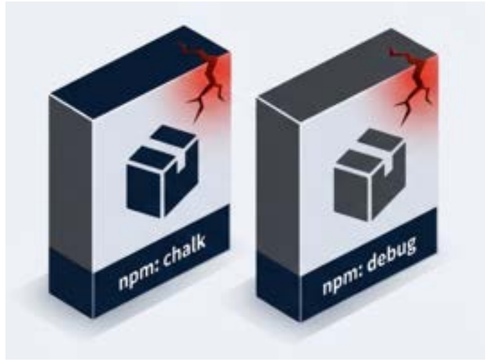
기밀 정보를 유출하고 생태계 전반에 걸쳐 스스로 복제하도록 설계(Worm)

DeepDive

01. 외부의 악성코드 위협 (Malware)

[Case 01]

debug, chalk 패키지 하이재킹



2025년 9월 8일경 발생한 인기 npm 패키지(debug, chalk 등을 포함하되 이에 국한되지 않음)를 대상으로 한 광범위한 npm 공급망 공격, 패키지 관리자 계정 탈취(ATO,Account Takeover) 를 통해 수천만의 다운로드 패키지 장악/업데이트에 악성 코드가 삽입

sonatype-2025-003716 / sonatype-2025-003727로 추적

대상

Debug, Chalk 및 16개 이상의 의존성 패키지

오픈 소스 패키지를 장악함으로써 기밀 정보를 탈취하고, 백도어를 심고 , 조직에 침투하는 APT 공격 기반 마련

원인

사회공학적 기법으로 메인테이너의 계정 탈취

관리자 계정의 권한을 탈취해 단번의 공격 시도로 npm 오픈소스 생태계에 영향을 끼침

흐름

메인테이너 계정 탈취 > 악성 코드 삽입 > 패키지 자동 배포 > 개발자 감염

악성버전이 유포되고 침해 사실이 확인될 때까지 약 2시간 이상 공격에 무방비 노출됨

영향 범위

chalk : 5.6.1, chalk-template : 1.1.1, color : 5.0.1, color-convert : 3.1 등 수많은 의존성 패키지 포함

이후에도 의존성 연관된 추가 피해 패키지 등장

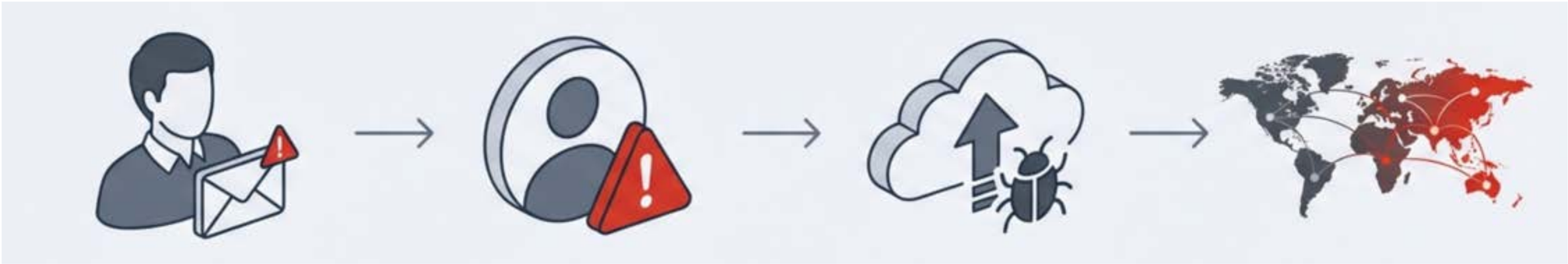
npm Chalk and Debug Packages Hit in Software Supply Chain Attack : Sonatype blog

DeepDive

[Case 01]

debug, chalk 패키지 하이재킹

전형적인 공급망 공격 패턴: 단 하나의 계정 탈취로 수백만 개발자에 도달



1. 사회 공학적 기법
공격자가 패키지 메인테이너를 대상으로 피싱 시도(MFA위장)

2. 계정 탈취
메인테이너의 계정 권환 획득

3. 악성 버전 게시
정상 업데이트로 위장하여 debug/chalk 등 타 패키지와 의존성이 높은 패키지에 악성 코드 배포

4. 다운스트림 감염
자동화된 빌드 시스템들이 최신 버전을 적용하며 확산

DeepDive

01. 외부의 악성코드 위협 (Malware)

[Case 02]

Shai-Hulud 2.0 캠페인

2025년 11월 24일경 시작되어 지금까지도 이어지고있는 npm 생태계 대상의 지능형지속위협 (APT, Advanced Persistent Threat)

sonatype-2025-003810로 추적

대상

NPM 생태계 전반

특정 타겟을 두고 발생한 위협이 아니라, 무차별 확산되는 특징을 가짐

타입

APT (지능형 지속 위협) + Self-replicating Worm (자가 복제)

단순 해킹이 아닌, 소프트웨어 공급망 내에서 스스로 생존하고 번식하도록 설계

흐름

악성 패키지 설치 및 스크립트 실행 > 주요 자격 증명 수집 탈취 > C&C 유출 > 피해자의 다른 패키지에 재배포해 확산

보안 사각지대를 의식한 탐지 우회 설계로 피해가 확산

영향 범위

airchief - 0.3.1, airpilot - 0.8.8, angulartics2 - 14.1.1, 14.1.2, browser-webdriver-downloader - 3.0.8 외에도 공동 메인테이너를 공유하는 수많은 패키지

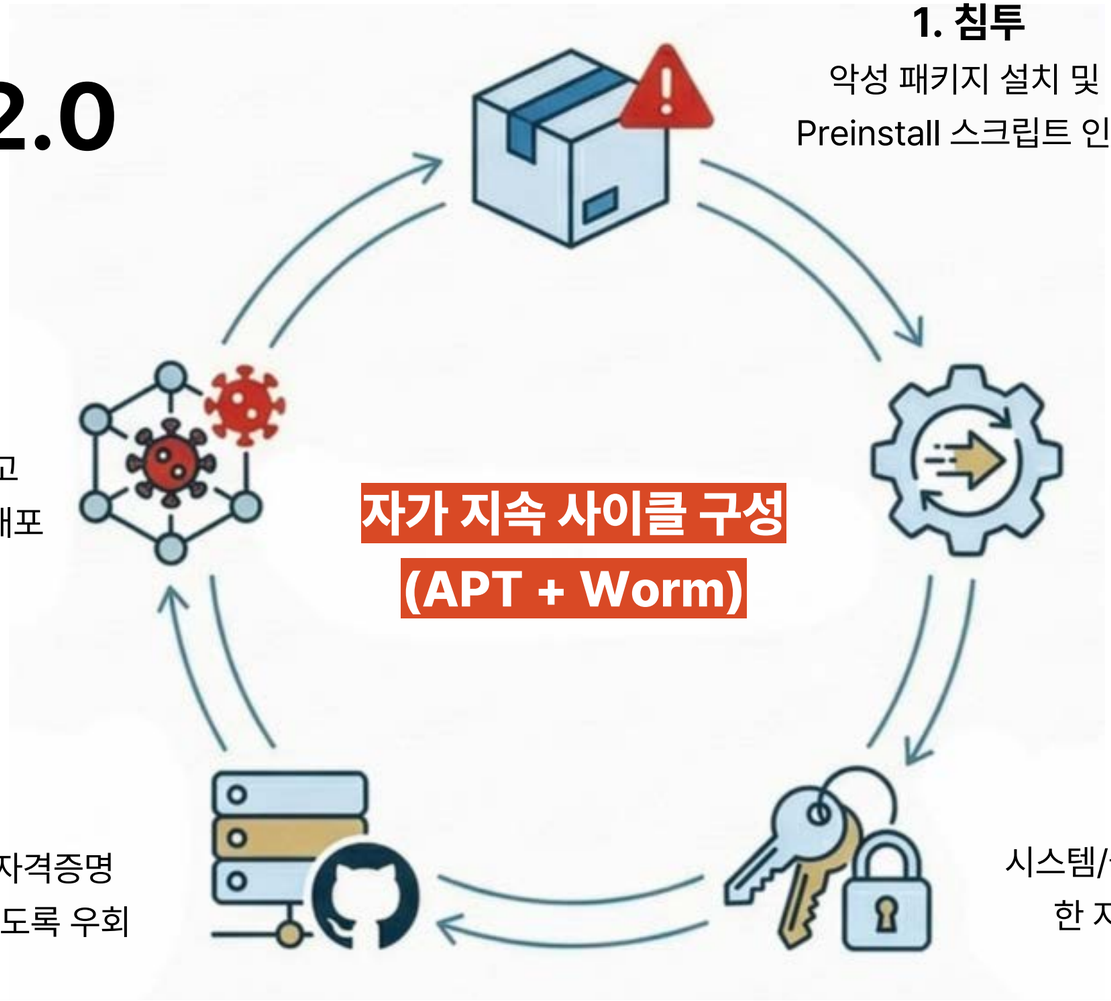
이후에도 의존성 연관되어 있지 않더라도 추가 피해가 계속 늘어나는 중

Ongoing npm Software Supply Chain Attack Exposes New Risks : Sonatype blog

DeepDive

[Case 02]

Shai-Hulud 2.0 캠페인



1. 침투
악성 패키지 설치 및
Preinstall 스크립트 인입

2.실행
Bun 런타임 설치로 탐지 우회

3. 탈취
시스템/클라우드/github 등 광범위
한 자격 증명을 탐색 및 수집

4. C&C 및 유출
Github public repo에 탈취한 자격증명
유출해 합법적인 트래픽처럼 보이도록 우회

5. 전파
피해자의 다른 repo가 있는지 찾고
다른 패키지에도 백도어를 심어 재배포

[Case 02]

Shai-Hulud 2.0 캠페인

```
# Process each repository
echo \"$REPOS_RESPONSE\" | jq -c '.[]' | while IFS= read -r repo; do
    REPO_NAME=$(echo \"$repo\" | jq -r '.name')
    REPO_OWNER=$(echo \"$repo\" | jq -r '.owner.login')
    REPO_FULL_NAME=$(echo \"$repo\" | jq -r '.full_name')
    DEFAULT_BRANCH=$(echo \"$repo\" | jq -r '.default_branch // \"main\"')

    echo \"📦 Processing repository: $REPO_FULL_NAME\"

    # Get the latest commit SHA from the default branch
    echo \"  → Getting default branch SHA...\"
    REF_RESPONSE=$(github_api GET \"/repos/$REPO_FULL_NAME/git/ref/heads/$DEFAULT_BRANCH\")
    BASE_SHA=$(echo \"$REF_RESPONSE\" | jq -r '.object.sha // empty')

    if [ -z \"$BASE_SHA\" ]; then
        echo -e \"  ${RED}❌ Could not get default branch SHA. Skipping...${NC}\"
        continue
    fi

    # Create new branch
    echo \"  → Creating branch: $BRANCH_NAME\"
    BRANCH_DATA=$(jq -n \\
        --arg ref \"refs/heads/$BRANCH_NAME\" \\
        --arg sha \"$BASE_SHA\" \\
        '{ref: $ref, sha: $sha}')
```

[Worm 작동 방식]

1 저장소 순회

감염된 환경에서 사용자 접근 권한이 있는 저장소 목록을 가져온 다음, while 루프를 사용하여 각 저장소를 순회

2 브랜치 생성

각 저장소에 대해 shai-hulud라는 이름의 새 브랜치를 생성

3 악성 워크플로우 업로드

github/workflows/shai-hulud-workflow.yml
(GitHub Actions 워크플로가 포함)이라는 파일을 이 새 브랜치에 업로드
워크플로가 실행되면 github public repo C&C에 비밀 정보가 유출

4 풀리퀘스트 생성

shai-hulud 브랜치를 저장소의 main 브랜치에 병합하는 PR(Pull Request)를 생성

Ongoing npm Software Supply Chain Attack Exposes New Risks : Sonatype blog

02. 내부의 취약점

Vulnerability

DeepDive

02. 내부의 취약점 (Vulnerability)

React2Shell

2025년 12월 3일에 공개된 RSC(React Server Components)의 인증이 필요없는 RCE(Remote Code Execution) 취약점을 일컫는 별칭으로, 심각성과 파급력으로 인해 보안 커뮤니티에서는 과거의 'Log4Shell'에 비유됨

CVE-2025-55182 / CVE-2025-66478 으로 관리

대상

React Server Components (Flight Protocol)

프로토콜 취약점

원인

안전하지 않은 역직렬화

서버가 클라이언트의 요청을 받아 메타데이터를 재구성하는 과정에서 검증 부족 발생해 RCE 가능성

흐름

공격자의 악의적인 HTTP 페이로드 전송 > 서버의 역직렬화기가 이를 검증 없이 처리 > 내부 모듈 접근 권한 노출 > 원격 코드 실행

보안 사각지대를 의식한 탐지 우회 설계로 피해가 확산

영향 범위

React 19.0.0~19.2.0 버전 및 이를 사용하는 Next.js(15.x, 16.x 등)를 포함한 다수의 현대적 웹 프레임워크가 전부 해당 / CVSS 10.0 최고점

RCE 이외 소스코드 노출(CVE-2025-55183) / 서비스 거부(Dos,CVE-2025-55184) 등 연쇄 취약점 발생

React2Shell 익스플로잇

1. 침투

사용자 계정 정보 /
엔드포인트 정보 없이

2. 프로토콜 취약점

조건없는 역직렬화

```
78
79    export function requireModule<T>(metadata: ClientReference<T>): T {
80      const moduleExports = parcelRequire(metadata[ID]);
81  -   return moduleExports[metadata[NAME]];
82    }
```

3. 검증되지 않은 메타 데이터 처리

악성 패키지 설치 및
Preinstall 스크립트 인입

4. 터미널 실행

RCE

DeepDive

React2Shell 위험성 분석

주요 위험 요소

인증 불필요

RCE(Remote Code Execution) 공격자가 유효한 계정 정보나 세션을 요구하지 않고 네트워크 접근 권한 만으로 공격을 시도할 수 있음

공개된 익스플로잇

취약점 공개 직후 PoC(Proof of Concept) 코드가 GitHub 등에 공개되어 기술적 숙련도가 낮은 공격자도 이를 악용할 수 있음

공격의 높은 성공률

특별히 제작된 HTTP 요청 하나만으로도 공격이 성공하며, 단 한번의 요청으로 침투 성공

기본 설정의 취약성

개발자가 서버 함수 엔드포인트를 명시적으로 노출하지 않았더라도, 취약한 버전의 RSC 로직이 포함된 React / Next.js를 사용한다면 공격 대상이 됨

DeepDive

React2Shell 공격 데모

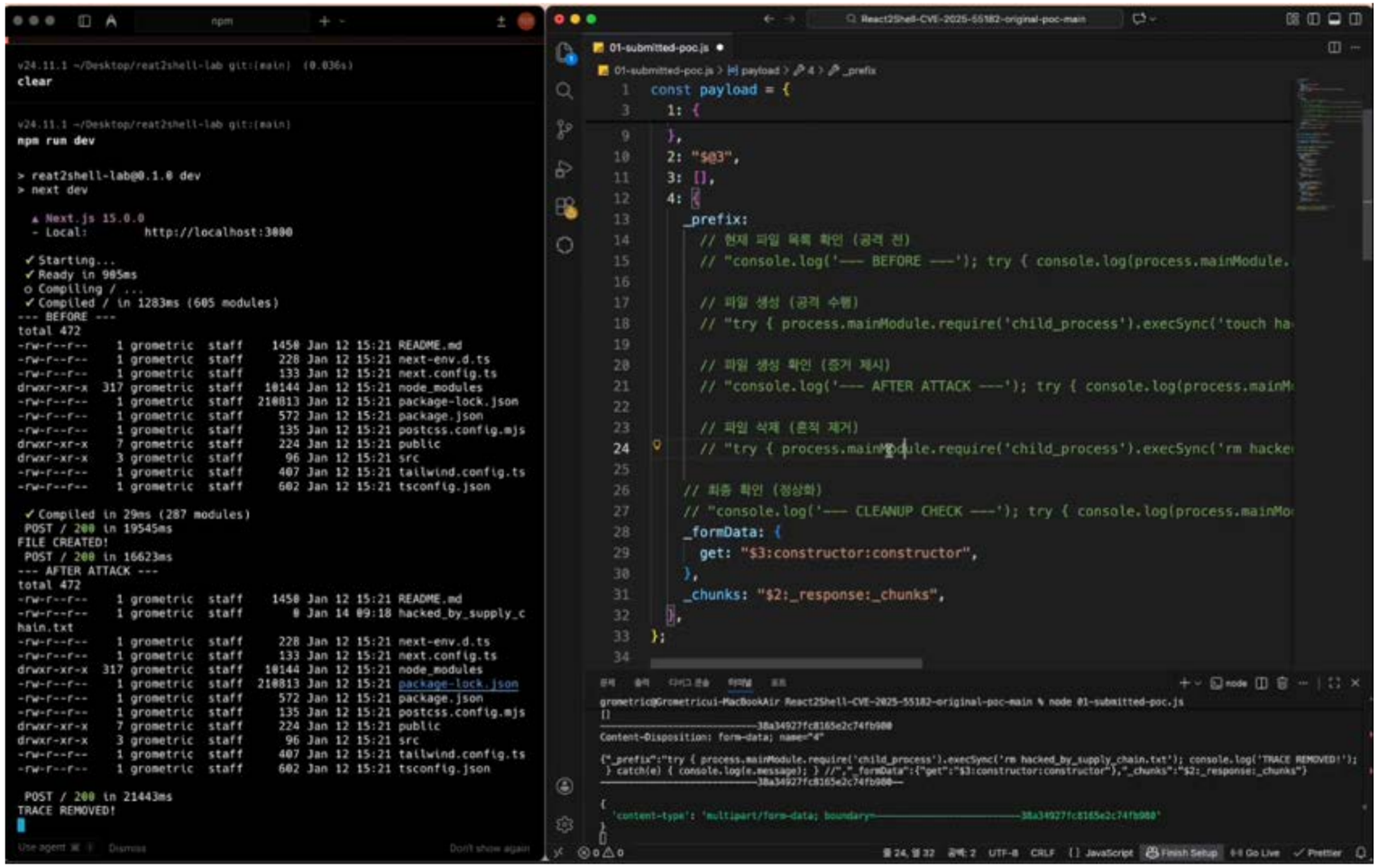
테스트 환경 명세

Victim)
node : 24.11.1
npm : 11.6.2
next.js : 15.0.0 (문제 취약점 버전)

Attacker)
node : 24.11.1
npm : 11.6.2

공격 시나리오

1.가해자가 피해자 환경에 접속
2. 피해 환경의 파일목록 확인
3. 피해 환경에 hacked_by_supply_chain.txt 생성
4. hacked_by_supply_chain.txt 파일 생성 확인
5. hacked_by_supply_chain.txt 삭제
6. hacked_by_supply_chain.txt 삭제 확인



Problem

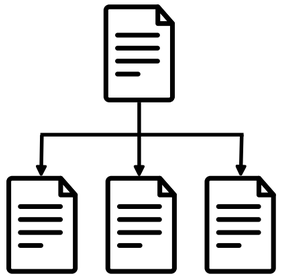
보안의 사각지대:

수동 검증의 현실적인 한계

Problem

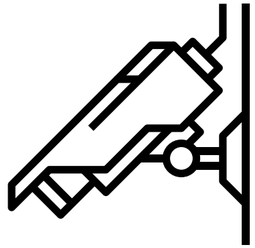
수동검증의 현실적인 한계

가시성 부재



- npm install 명령어 한줄로 수백개의 패키지가 유출
- 직접 의존성보다 간접 의존성이 80-90% 이상 차지
- 개발자나 보안팀이 수동으로 코드를 리뷰하거나 방대한 트리를 검증하는 것은 물리적으로 불가능

탐지 사각지대



- 작년 하반기 Malware이슈 급증
- CVE 기반 분석 결과는 전방위의 보안 위협을 대처하지 못함
- CVE, Malware를 별개의 인텔리전스로 관리하고 업데이트 해야 실시간 위협에 대응할 수 있음

Problem

수동검증의 현실적인 한계

자산 중앙화

sonatype nexus repository

능동 / 자동화 방어

sonatype repository firewall

가시성 부재

탐지 사각지대

사고 후 패치(Reactive)에서

'반입 전 차단(Proactive)으로의 패러다임 변화'

Solution

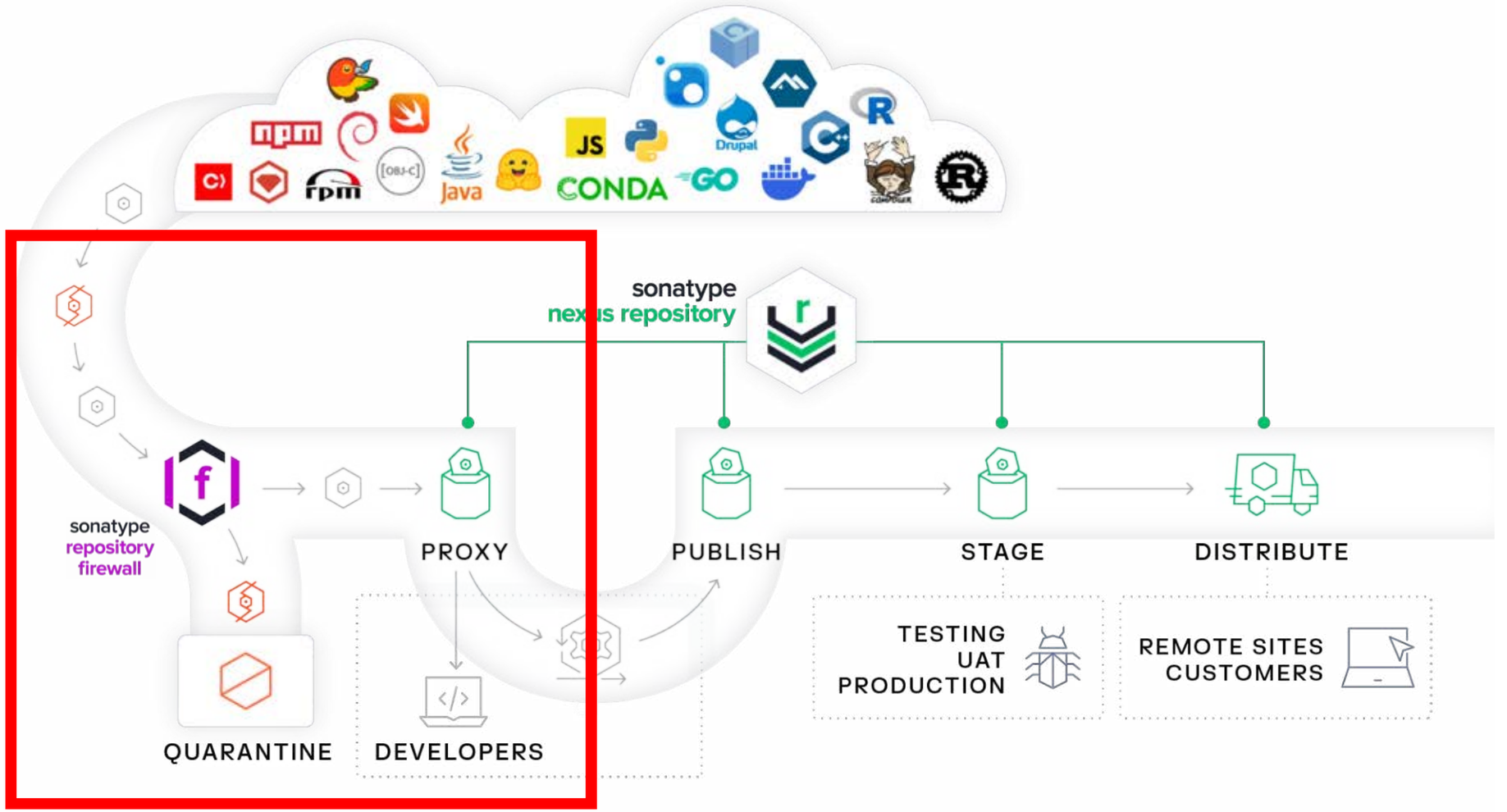
Nexus Repository & Repository Firewall로 완성하는:

자동화 방어 체계

Solution

Sonatype으로 구축한 자동화 방어 체계

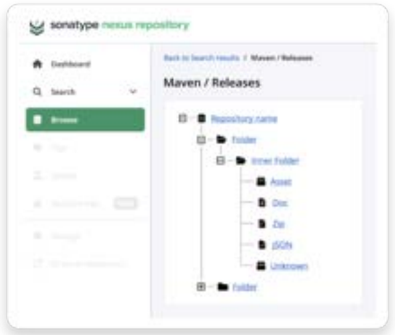
사고 후 패치(Reactive)에서 반입 전 차단(Proactive)으로의 패러다임 변화



Solution

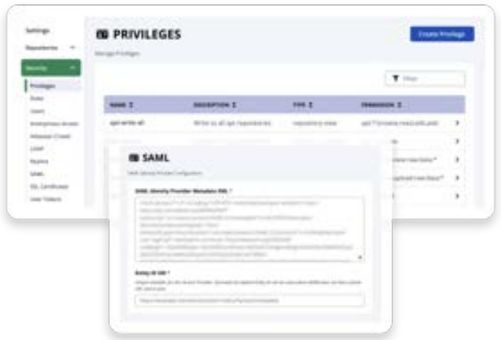
Sonatype Nexus Repository Pro

1) 통합 아티팩트 관리



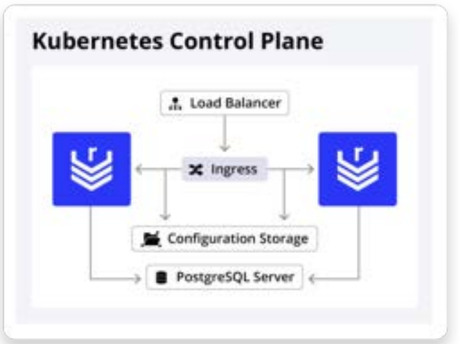
다양한 유형의 바이너리 및 아티팩트를 단일 저장소에서 중앙 집중식으로 관리하여 관리 효율성을 극대화 (Maven, npm, Docker, PyPI 뿐만 아니라 Hugging Face 등 주요 패키지 포맷지원)

2) 엔터프라이즈 급 보안



역할 기반 접근 제어(RBAC), SAML/SSO 지원을 통한 사용자 관리 강화 및 TLS 암호화 통신 및 철저한 감사 로그(Audit Logs) 기록을 통해 권한 통제

3) 고가용성 및 성능 최적화

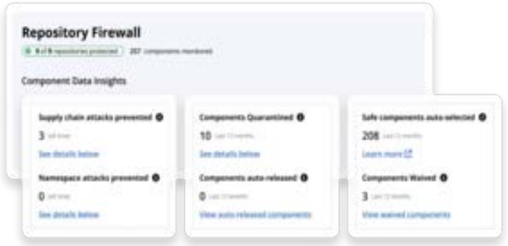


스마트 프록싱(Smart Proxying) 및 로컬 캐싱을 통해 대기 시간 최대 95% 감소하고, 고가용성(HA) 클러스터 및 에지 노드 구성 가능

Solution

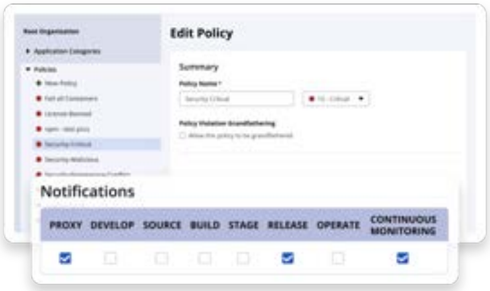
Sonatype Repository Firewall

1) 포괄적인 멀웨어 인텔리전스



Sonatype의 독자적인 AI 및 업계 최고의 멀웨어 인텔리전스 활용하여 소프트웨어 개발생명주기 전반을 보호

2) 맞춤형 컴포넌트 제어

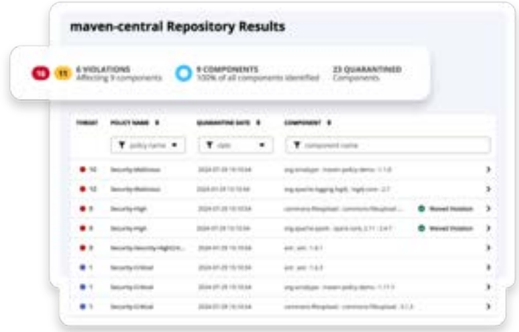


네크워크 단계에서 악성 오픈소스를 선제적 차단

승인되지 않은 경로로 유입되는 악성 코드 위험 완화하고, 보안 사고 발생전 차단을 통해 사후 수습 비용 효율화

격리된 컴포넌트가 안전한 것으로 확인되면 자동으로 격리 해제

3) 악성 오픈소스 자동 격리 시스템



조직의 보안 정책, 라이선스 규정, 품질 표준을 자동으로 강제하여 거버넌스를 확립하고 각 단계별 세부 적용 커스터마이징 가능

Takeaways

최소한의 즉각적인 대응 가이드

Nexus Repository 커뮤니티 버전으로 구축하는 최소한의 방어선

* Nexus Repo Pro를 사용하면 이중화, 컨텐츠 복제, rbac접근제어, 권한별 업로드등, 특정패턴의 요청 등 차단, 무결성검증 등 추가 기능도 사용할 수 있습니다

1 모든 개발/배포 트래픽의 중앙화

Public Repository(NPM, Maven 등)으로 나가는 연결을 차단하고, Nexus Proxy Repository를 경유하도록 설정

2 문제 패키지의 수동 삭제

Nexus Proxy Repository 관리자 화면에서 해당 패키지 버전을 즉시 삭제하여 접근 차단

3 검증된 버전만 캐싱

Nexus Repository에는 안전하다고 판단된 패키지만 다운받고, 다운로드된 패키지를 내부에 캐싱해 클린버전으로 안정적인 빌드 유지

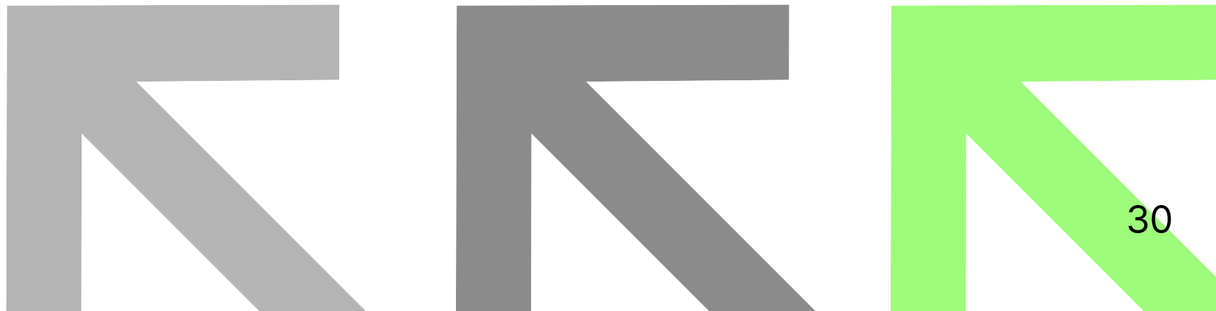
3rd GroMeetUp

Session 02

Hugging Face - Nexus Repository 연동

Enterprise AI의 안전한 항해법

GroMetric
김관영



목 차

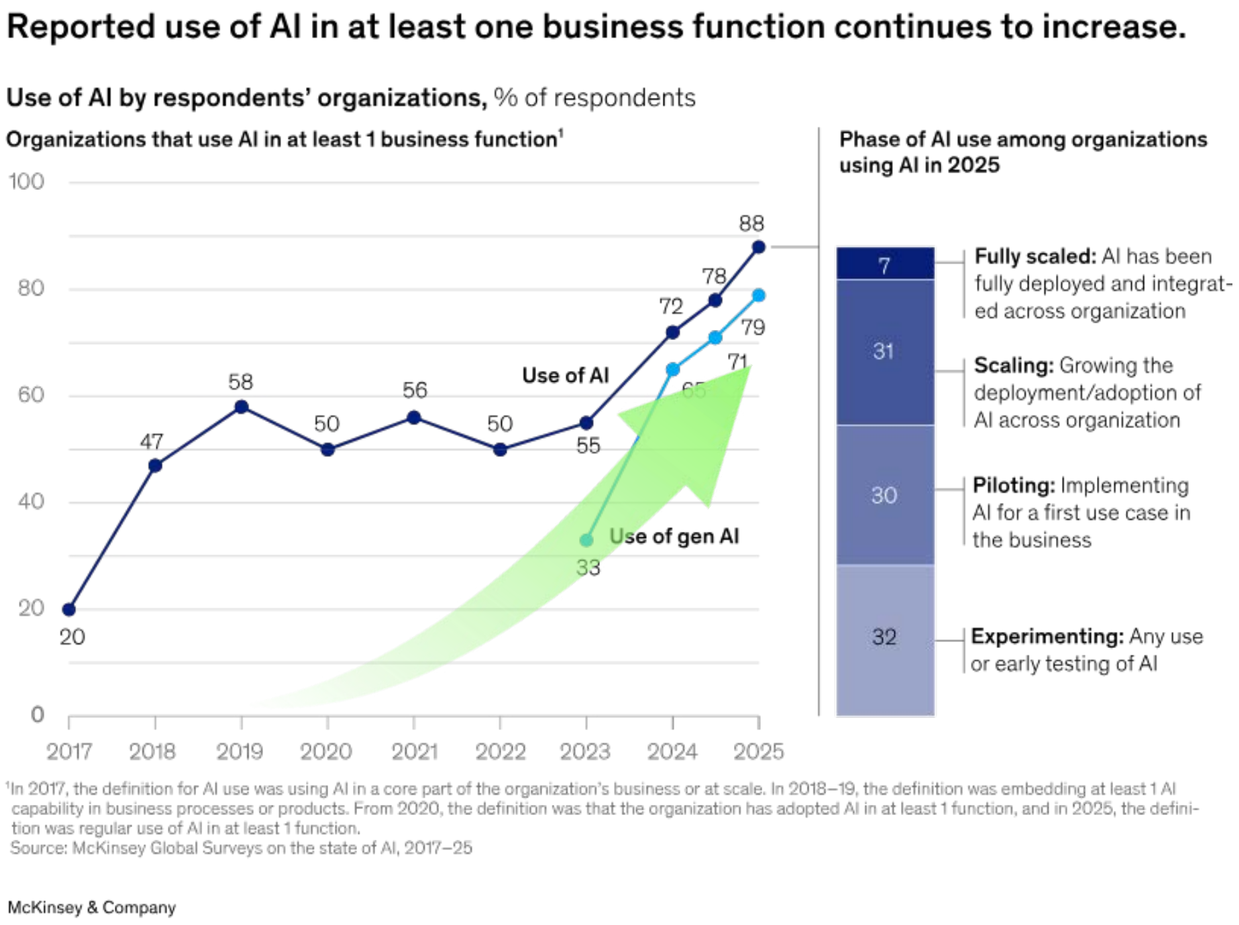
Target

AI 혁신의 허브

Hugging face와 그림자

Target

AI 사용률 급증과 폭발적인 성장



글로벌 기업 AI 도입률 88% 돌파

단순 '실험'에서 경영 '필수재'로 안착

기업 62% 도입 및 검증 착수

자율 수행 가능한 '에이전트(Agent) AI' 부상

실질적 성과 창출 집중

파일럿 단계를 넘어 워크플로우 재설계 및 전사적 확산

AI 사용률 급증

AI 시장 폭발적인 성장

https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai

Target

AI 혁신의 허브, Hugging face와 그림자

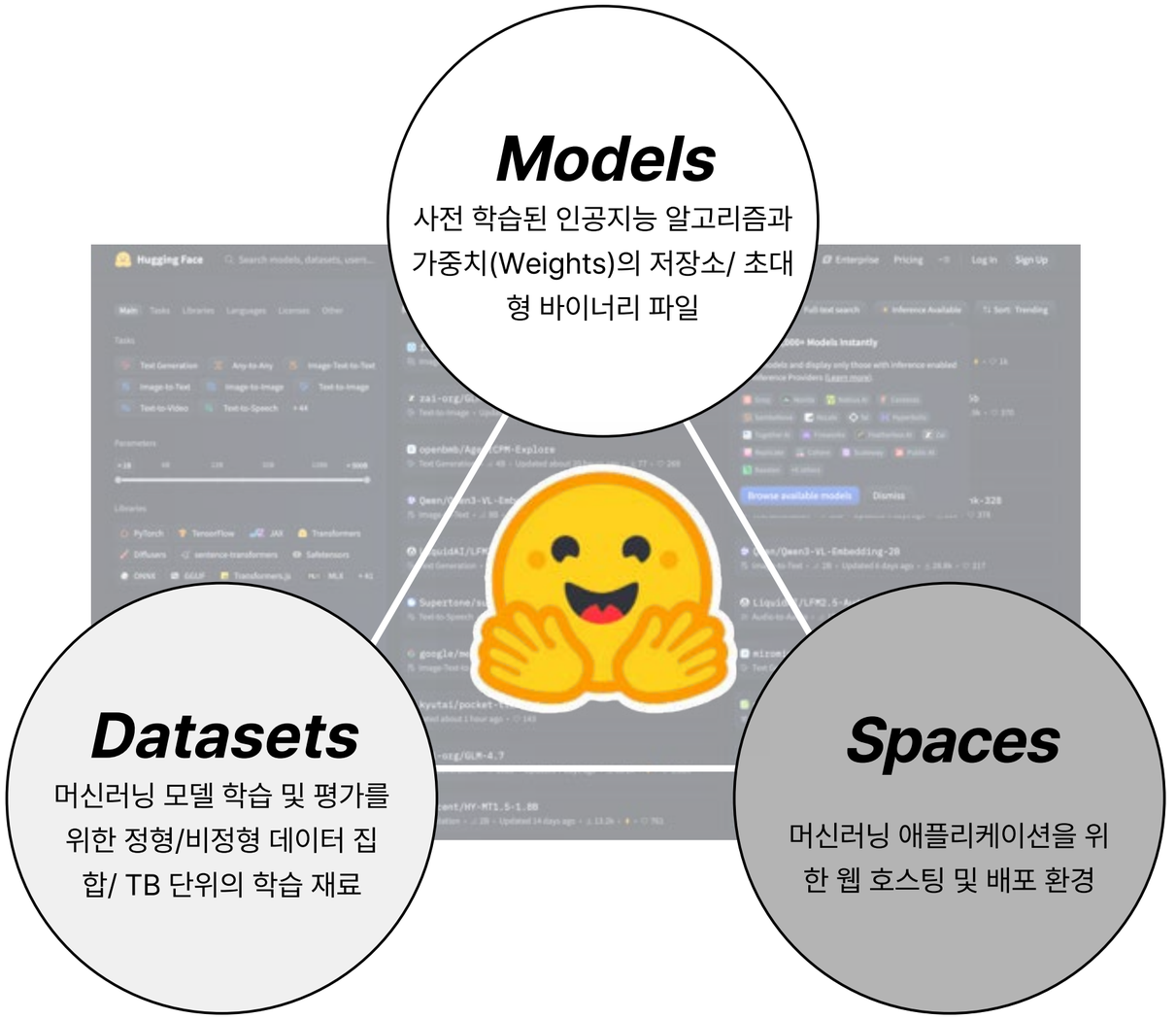
Hugging Face

1 AI 오픈소스 허브

머신러닝계의 깃허브(GitHub)로, 전세계 AI 모델/데이터셋/AIapp을 모두 제공하는 오픈소스 플랫폼으로 240만개 이상의 모델, 70만개 이상의 데이터 셋 보유

2 사용자 및 조직 규모

약 500만명의 등록 사용자, 30만개 이상의 조직, 10,000개 이상의 기업이 공식 활용, 7,774개의 검증된 기업 계정이 등록됨
누구나 업로드가능한 열린 플랫폼으로
다양한 포맷(PyTorch, Pickle, Safetensors, GGUF 등)을 지원 및 유통



Target

AI 혁신의 허브, Hugging face와 그림자

새로운 공격 표면으로 급부상

Hugging Face 혁신의 그림자

AI 개발 필수요소

Hugging Face에서 검증된 모델 (Llama, BERT 등)을 다운로드 받아 튜닝하는 것이 표준

무방비한 연결

누구나 모델을 업로드할 수 있고 누구나 쉽게 다운로드 받을 수 있기 때문에 악성코드가 기업 공급망으로 유입되는 새로운 통로가 되고 있음

AI 보안 이슈 급증

CVE-2025-14925
(Hugging Face Accelerate RCE)
CVE-2025-14928
(Transformers 라이브러리 RCE)
등 AI 보안 이슈 대두

Key Risk

인프라 / 거버넌스 / 위협

엔터프라이즈 AI 보안의 10가지 취약점과 핵심 과제 3가지

Key Risk

엔터프라이즈 AI 보안의 10가지 취약점과 핵심 과제 3가지

OWASP LLM Top 10

LLM04: 데이터 및 모델 오염 (Data and Model Poisoning)

모델의 학습, 미세 조정(Fine-tuning), 데이터 조작으로 백도어나 편향 주입
공격자가 학습 데이터에 악의적인 내용을 넣어 모델이 특정 상황에서 공격자가 원하는 방향으로 조작

위험: 모델 성능 저하, 유해한 콘텐츠 생성, 숨겨진 명령 실행(백도어)

LLM03:2025 공급망 (Supply Chain) 취약점

데이터, 모델, 플러그인 등 외부 구성 요소에서 발생하는 취약점
사전 학습된 모델에 백도어나 라이브러리의 취약점, 오염된 데이터셋 사용 등

위험: 시스템 전체의 무결성 훼손, 편향된 결과 출력, 시스템 장악

1 프롬프트 인젝션

2. 민감 정보 노출

3. 공급망 취약점

4.데이터 및 모델 오염

5.부적절한 출력 처리

6.과도한 위임

7.시스템 프롬프트 유출

8.벡터 및 임베딩 취약점

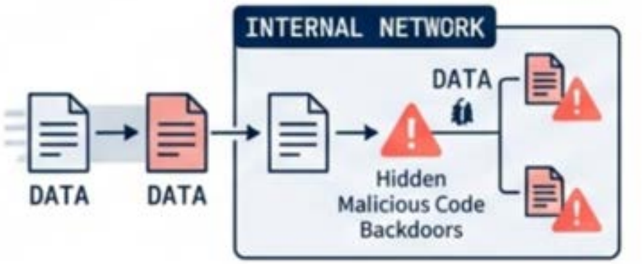
9.허위 정보

10.제한 없는 리소스 소비

Key Risk

엔터프라이즈 AI 보안의 10가지 취약점과 핵심 과제 3가지

핵심 과제 세 가지



대용량 모델 전송 병목

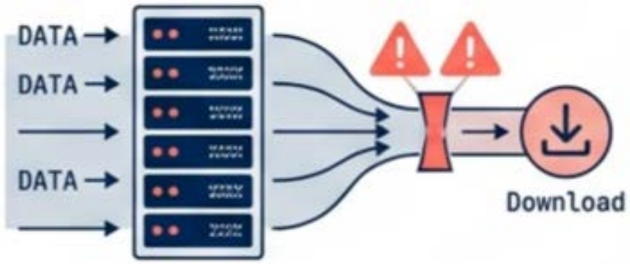
대규모 AI 리소스 요청으로 인해 서비스 장애(DoS)를 유발하거나 과도한 인프라 비용 및 부하를 발생시키는 리스크

1. 인프라

AI 변이와 라이선스 위반

외부 모델을 활용하는 과정에서 출처가 불분명하거나 정책에 위반되는 요소를 수용하게 되는 리스크

2. 거버넌스



데이터로 위장한 실행 파일, 악성모델 반입

공격자가 모델 파일 자체를 조작(Poisoning)하여 백도어나 악성 코드를 주입하는 위협

3. 위협

Risk 01

Infra: 대용량 모델 전송 병목

대용량 모델 다운로드 병목현상 발생

테스트용 Llama3 소형 8B

14.9GB

파라미터 수: 80억 개 (8,000,000,000)
숫자 1개 크기: 2 Byte (FP16 기준)
총 용량 (Byte): 16,000,000,000 Byte

서비스용 Llama3 중형 70B

130.4GB

파라미터 수: 700억 개 ($70,000,000,000$)
숫자 1개 크기: 2 Byte (FP16 반정밀도 표준)
총 용량 (Byte):140,000,000,000 Byte

> **Llama 3 70B 모델 1개를 다운로드하기 위해 최소 130GB의 네트워크 대역폭(Bandwidth)을 점유**

BitTorrent를 사용해 다운로드 속도 조정

https://ai.meta.com/blog/meta-llama-3/

hugging face 사용자 커뮤니티 / reddit 커뮤니티 의견

Risk 02

Governance: AI 변이와 라이선스 위험

AI 변이(AI mutation)란?

데이터셋에서 모델로, 다시 애플리케이션으로 이어지는 과정에서 상위 라이선스 의무가 누락되는 현상이 심각

[주요 라이선스 위협 유형]

1) AI 특화 라이선스(RAIL, Llama) 부족:
OpenRAIL: 군사적 목적이나 허위 정보 생성 등 특정 용도의 사용을 금지

Llama: 월간 활성 사용자(MAU) 7억 명 초과 시 별도 계약을 요구하며, 모델 출력을 경쟁 모델 학습에 사용하는 것을 금지

2) 라이선스 세탁:
상위 단계의 '상업적 이용 불가 (Non-Commercial)' 조항이 제거된 것을 인지하지 못하고 상용 제품에 통합할 경우 저작권 침해 및 소송 리스크가 발생

Open Source AI

메타(Meta)가 특정한 비상업적 라이선스로 LLaMA를 출시함

Fine-Tuning

제3자가 N차 가공을 거치는 과정에서 원본 라이선스 정보가 희석되거나 떨어져 나감

잘못된 라이선스 정보로 hugginface에 재배포됨

Hidden Risk

숨겨진 라이선스 독소 조항이나 출처의 불분명하여, 기존 도구로 탐지 실패

결국 원본 라이선스를 위반하게 됨

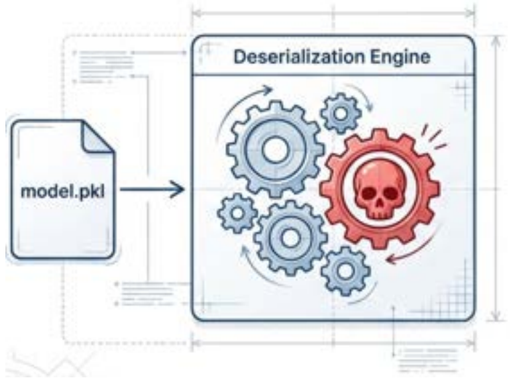
Hugging Face 모델 상당수가 'Unknown' 또는 'Other' 라이선스 태그 사용

명시적이지 않은 모델을 제외/격리해 리스크를 줄여야 함

Risk 03

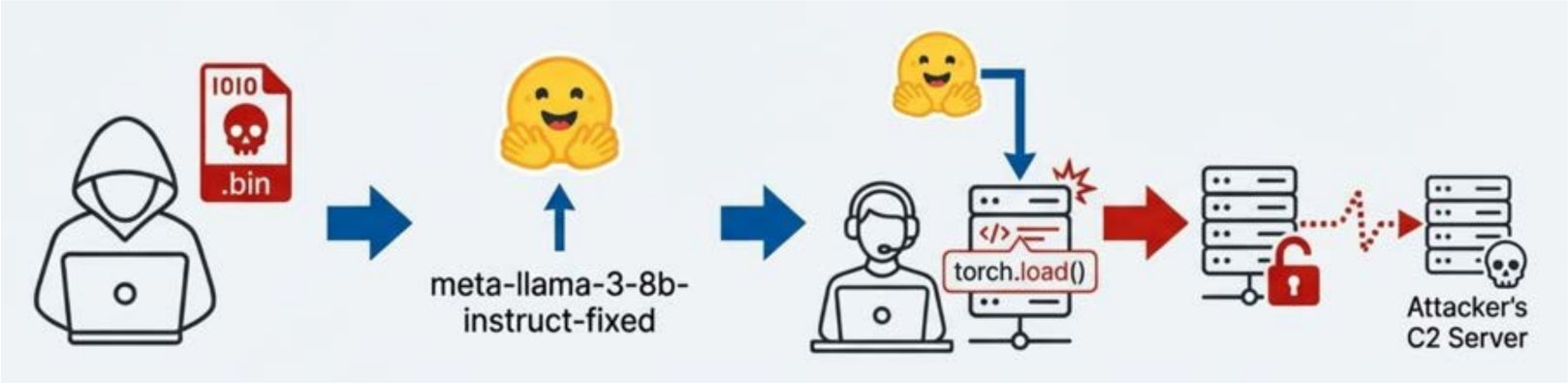
Threat: 악성 모델 반입

Pickle 역직렬화의 구조적 결함



모델이 로드되는 순간(torch.load() 등)
os.system이나 subprocess 명령어가 즉시 실행되도록 악성 페이로드를 주입할 수 있음

'데이터'로 인식하고 로드하지만, 실제로는 출처를 알 수 없는 실행 파일(.exe)을 실행하는 것과 동일한 위험에 노출



악성 모델 제작

공격자가 정상 모델의 . bin 파일에 리버스 셸(reverse shell) 페이로드를 삽입합니다.

Huggingface 배포

신뢰할 인기 모델 이름과 유사하게 위장하여 업로드
키워드 태그 등 추가해 노출

개발자 다운로드 및 실행

개발자가 의심없이 다운로드
pickle.load() 호출 실행

시스템 장악

Pickle 포맷(.pt, .bin, .pkl)이 본질적으로 안전하지 않음

약 44.9%가 여전히 안전하지 않은 Pickle 포맷을 사용 (높은 의존도)

Solution

솔루션을 활용한

중앙통제 관리 및 보안 자동화

Solution

중앙통제 관리 및 보안 자동화

Sonatype Nexus Repository & Firewall

Hugging Face의 Proxy 역할 및
로컬 캐싱을 통한 속도 보장

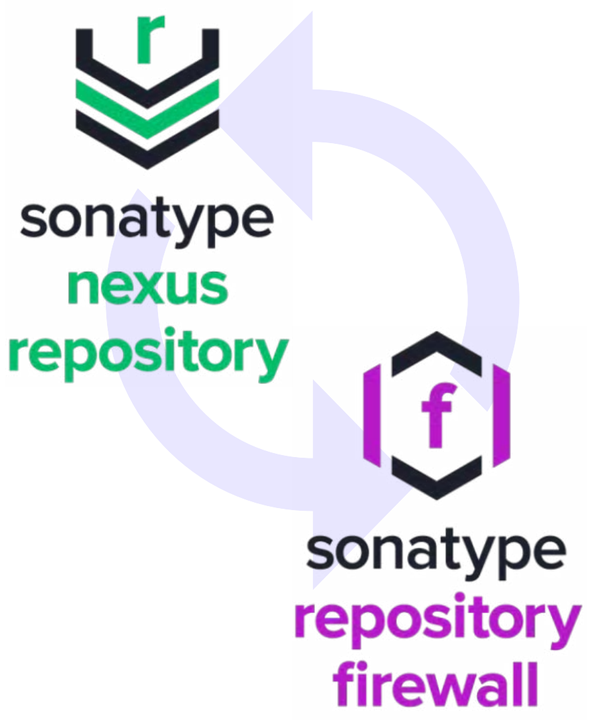
모델 유입 단계에서 악성 여부 및
라이선스 정책을 자동 검사하여 차단

검증된 모델만, 로컬속도로, 안전하게
공급하는 파이프라인 구축

Solution

Sonatype의 해결 방식

Nexus Repository & Firewall Ecosystem



Infra

다운로드 속도 향상 및 검증된 모델을 캐싱하고 개발지속
프록시 강제, 외부 접근 차단해 내부 저장소로 단일화

Governace

AI mutation 식별과 출처 관리를 통한 가시성 확보
정책 기반 거버넌스 자동화

Threat

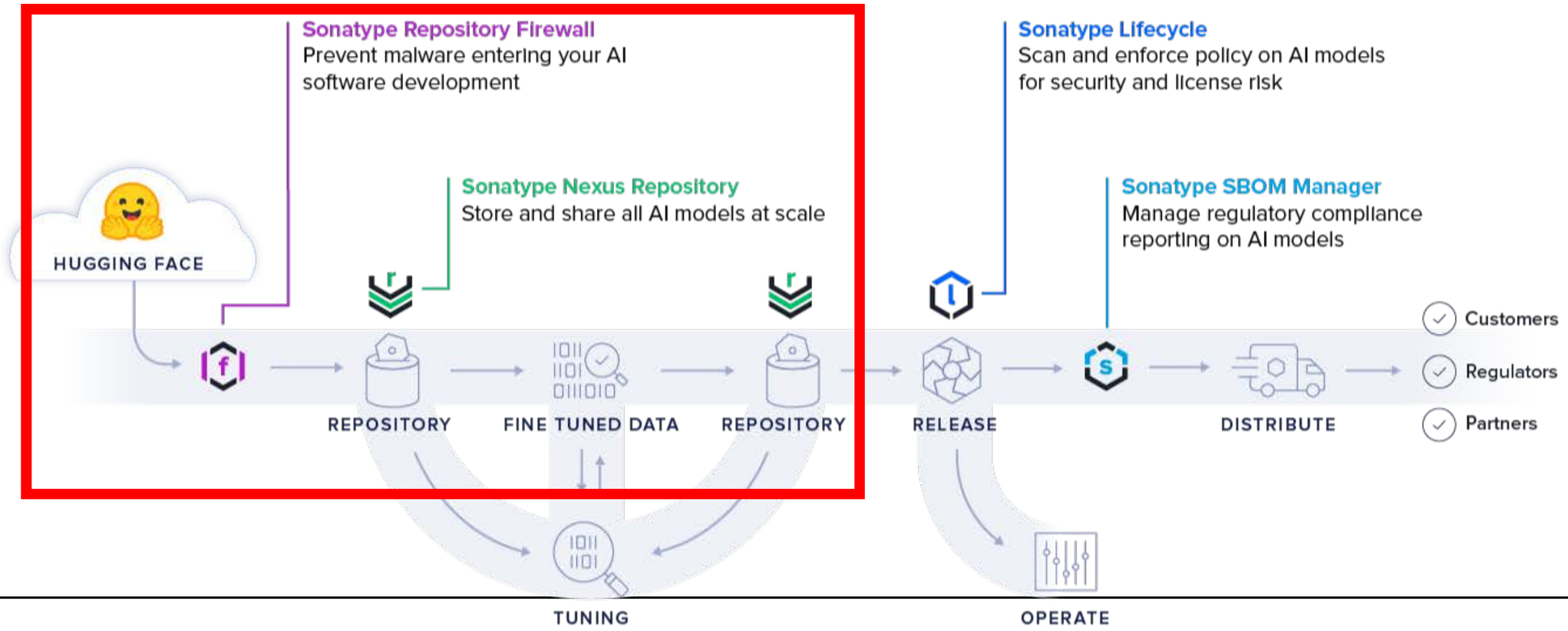
Malware 데이터베이스와 일치하는 모델을 즉시 차단 및 격리하고 위협 보고
소나타입 자체 AI 분석 DB를 실시간으로 반영

Solution

중앙통제 관리 및 보안 자동화

AI/ML Software Supply Chain

AI 개발은 한 번으로 끝나지 않으며, 모델을 가져와서(Proxy), 우리 데이터를 섞어 튜닝하고(Tuning), 다시 저장하는(Hosted)
무한 반복 과정을 단일 플랫폼에서 끊김 없이, 보안을 유지하며 수행

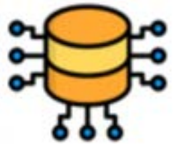
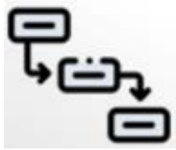


Solution

AI 투명성 확보를 위한 접근 방향

AIBOM

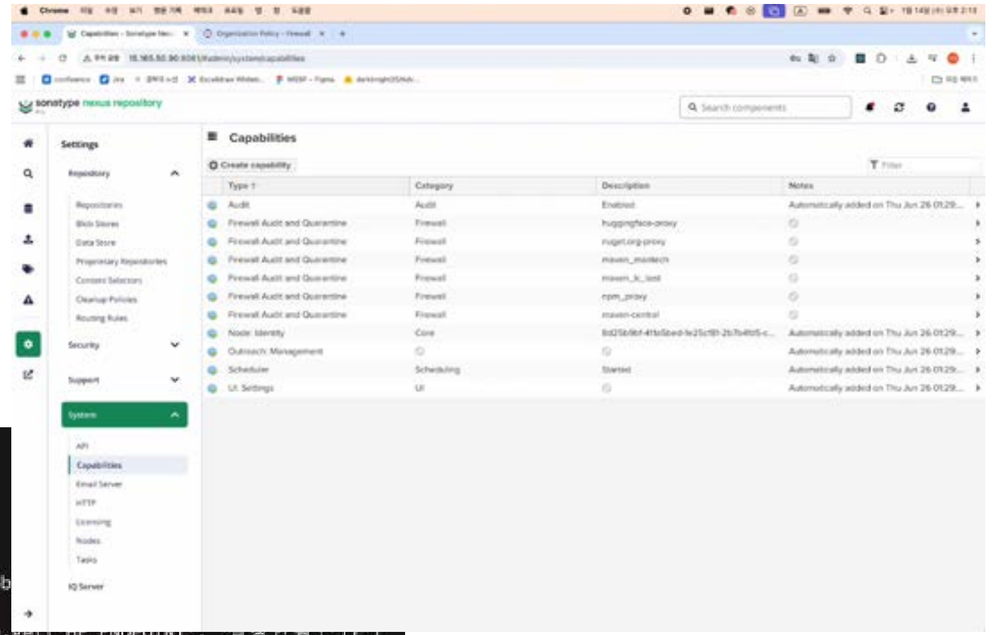
AI/ML 시스템을 구성하는 모든 요소를 포함하는 상세목록으로 AI 시스템에 특화된 데이터

알고리즘	의존성	인프라	데이터 소스	메타데이터
				
학습 Model, 고유(독점) 알고리즘, 오픈 소스 프레임워크는 특화된 AI 시스템의 핵심. 투명성을 유지하고 위험을 완화하기 위해서는 알고리즘의 출처, 라이선스 및 버전 관리 등 문서화 필요	라이브러리, 프레임워크 및 기타 소프트웨어 구성 요소로 TensorFlow, PyTorch, 랭체인 등이 포함됨	클라우드 서비스, GPU 및 기타 컴퓨팅 리소스는 AI 시스템 작동에 필수적인 요소로 시스템 성능, 확장성 및 비용에 영향	학습 데이터셋, 검증 데이터셋, 테스트 데이터셋은 AI 시스템의 기본 요소로 데이터셋의 품질, 출처 및 윤리적 고려 사항은 AI 모델의 정확성과 공정성에 중요	모든 구성 요소의 버전, 라이선스, 출처 및 기타 속성에 대한 정보 포함. 메타데이터를 통해 모든 이해 관계자는 감사 및 문제 해결에 필요한 세부 정보에 접근

Demo

Huggingface 연동 및 정책 기반 자동 격리 확인

Huggingface 연동



테스트 환경

Ubuntu 환경에서 python 스크립트 실행

테스트 모델

AndersGiovanni/gemma-2b(malware)

```
import os
from huggingface_hub import snapshot_download
from requests.exceptions import HTTPError

# 테스트할 모델과 리비전 (악성 의심 모델)
MODEL_ID = "AndersGiovanni/gemma-2b"
REVISION = "59ee2ac2c11d110710fb11ca58cc6669d5b

print(f"[*] 현재 HF_ENDPOINT 설정: {os.environ.get('HF_ENDPOINT', '설정되지 않음')}")
print(f"[*] '{MODEL_ID}' (Revision: {REVISION}) 다운로드 시도 중...")

try:
    # 다운로드 시도
    path = snapshot_download(repo_id=MODEL_ID, revision=REVISION)
    print("\n[FAILED] 다운로드가 성공해버렸습니다. (차단 실패)")
    print(f"저장 경로: {path}")
except HTTPError as e:
    # 403 Forbidden 에러가 뜨면 성공
    if e.response.status_code == 403:
        print("\n[SUCCESS] Nexus가 다운로드를 차단했습니다! (403 Forbidden)")
        print(f"에러 메시지: {e}")
    else:
        print(f"\n[ERROR] 예상치 못한 에러 발생: {e}")
except Exception as e:
    print(f"\n[ERROR] 실행 중 오류 발생: {e}")
ubuntu@ip-172-31-35-173:~$
```

절차

1. 모델 다운로드를 위한 python 스크립트 준비
2. Proxy repository 생성
3. 새로 만든 Proxy repository와 Firewall 연결
4. 기본 제공되는 Security-Malicious 정책 활성화
5. 스크립트 실행해 격리 여부 확인

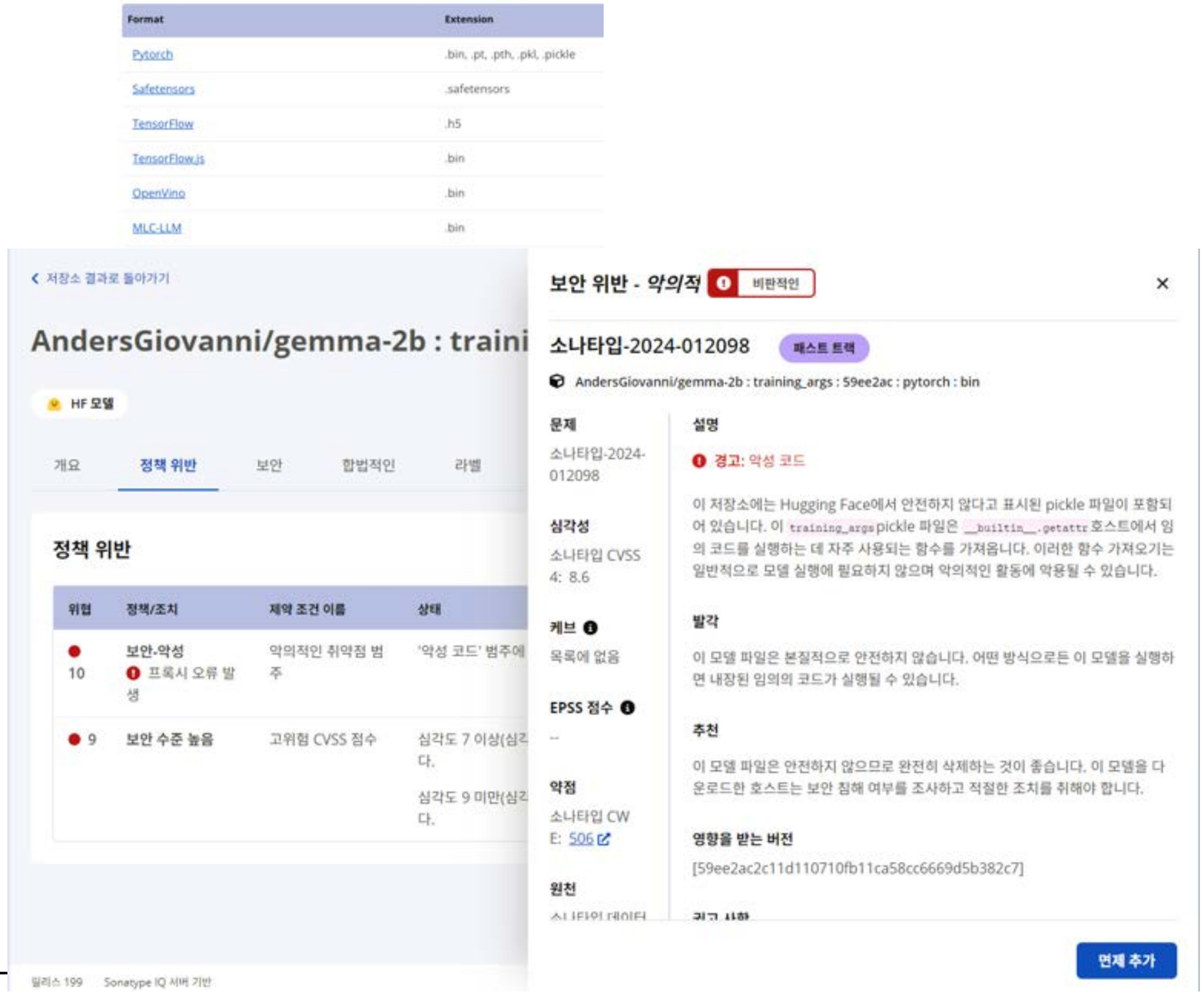
Demo

Huggingface 연동 및 정책 기반 자동 격리 확인

Huggingface - Nexus Repository 연동

지원되는 항목

hugging face의 모든
모델 포맷을 지원



https://help.sonatype.com/en/hugging-face-model-analysis.html