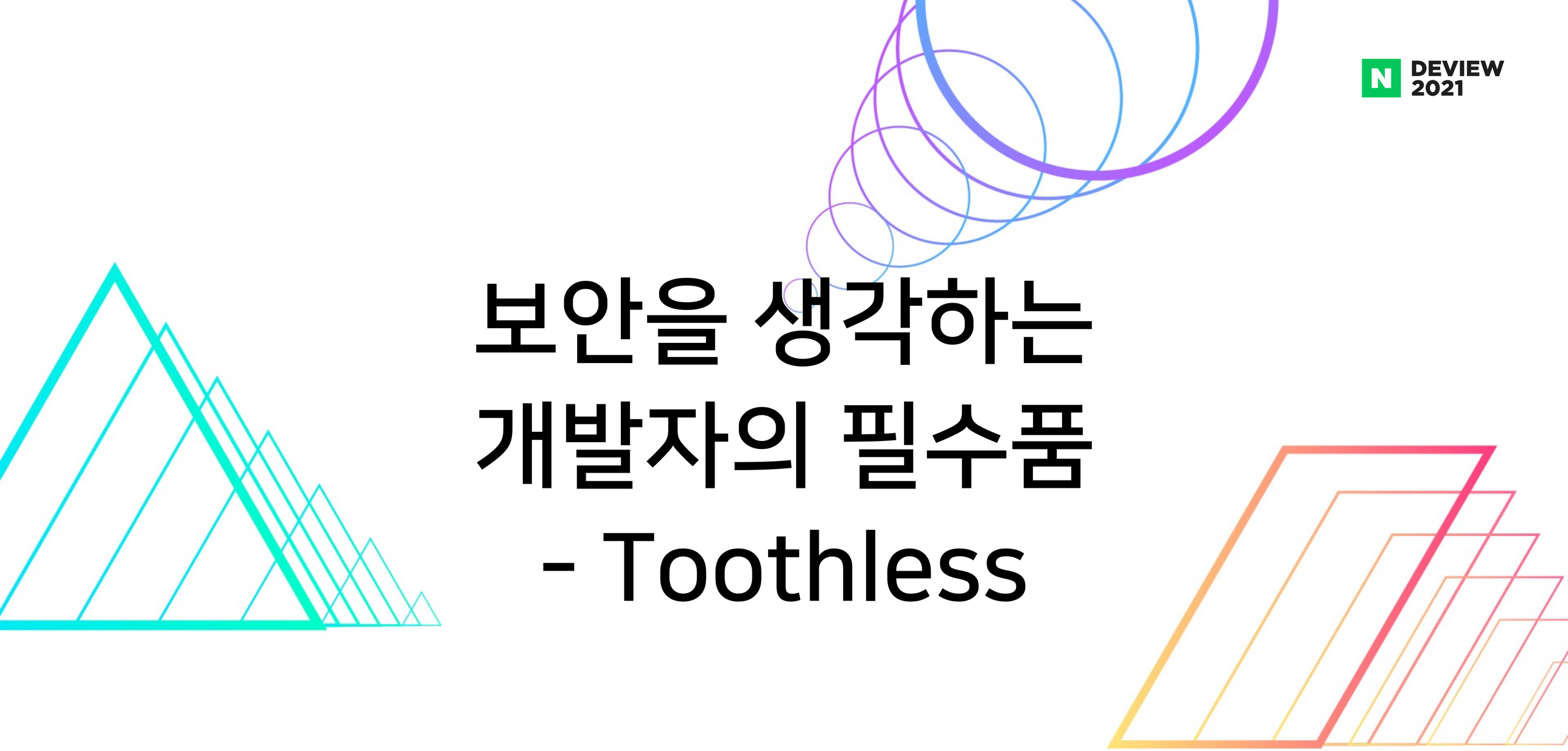


보안을 생각하는 개발자의 필수품 - Toothless

정연희 NAVER Security

CONTENTS

1. DevSecOps

1.1 Security Risks

올 한 해 발굴된 취약점은 17,447개…4년 연속 기록 갱신

좋아요 4개 | 입력 : 2020-12-17 20:18

작년에 비해 약 150개 늘어…위험도로 분류했을 때 비율은 거의 비슷

서두르는 개발 문화와 취약점 발굴 능력 향상…둘 다 취약점 증가에 영향

[보안뉴스 문가용 기자] 미국의 침해대응센터(US-CERT)가 올해 등록된 취약점의 수가 17,447개라고 발표했다. 4년 연속 기록이 갱신되고 있다. 작년의 취약점 수는 17,306개였다. 올해의 경우 고위험군 4,168개, 중간급 10,710개, 저위험군 2,569개인 것으로 집계됐다. 작년의 비율도 비슷하다.

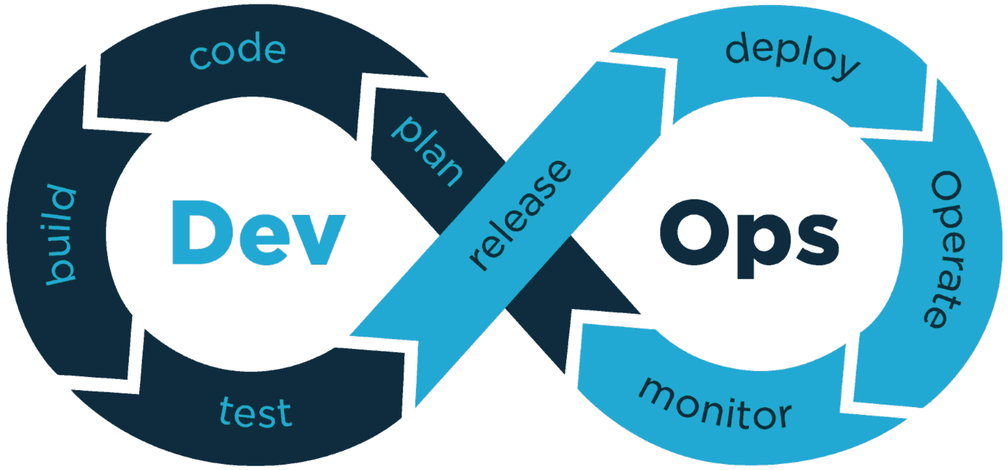
(https://www.boannews.com/media/view.asp?idx=93518)

1.2 DevOps

개발(Development) + 운영(Operation) = DevOps

- Cross Functional Team
- Widely Shared Metrics
- Automating Repetitive Tasks
- Post Mortems
- Regular Release

How to secure DevOps?



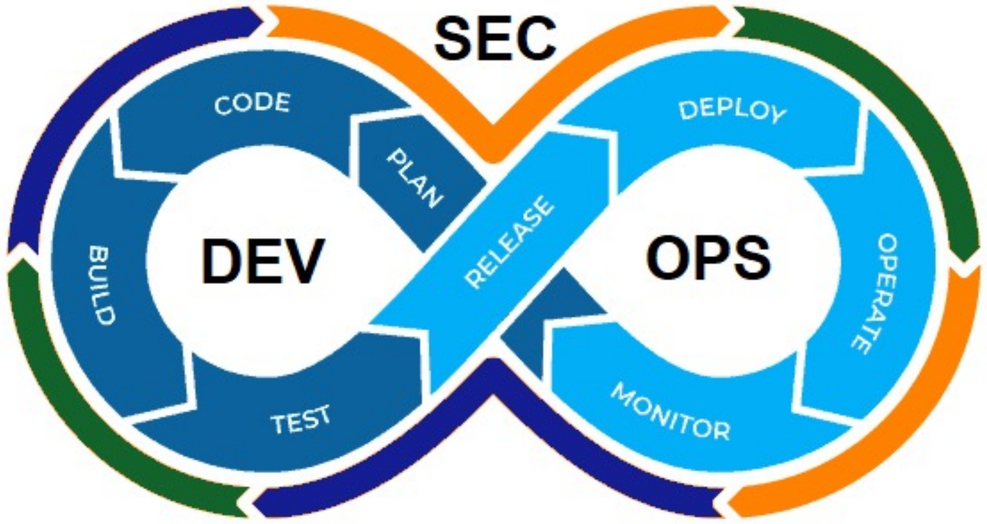
https://algoteque.com/jobs/devops-engineer/

1.3 DevSecOps

DEVIEW 2021

DevOps + 보안(Security) = DevSecOps

- DevOps의 전 영역이 보안과 연계됨을 의미
- 설계부터 개발, 테스트, 운영까지 SW lifecycle 전반에 보안을 통합
- 보안 리스크 최소화, 컴플라이언스 비용 절감과 같은 효과를 얻을 수 있음



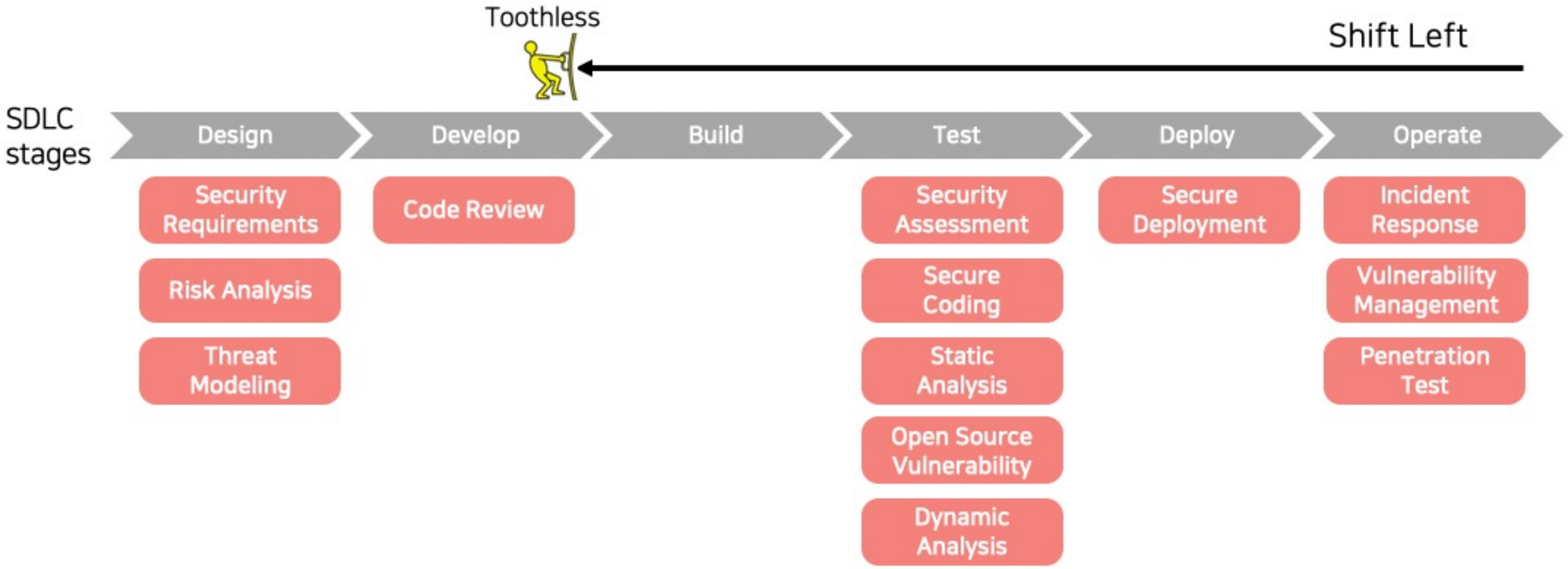
https://geekflare.com/what-is-sysops-dataops-devops-aiops-itops/

1.4 Let's Shift Left!

DEVIEW 2021

Shift Left

- 보안관련 Stage를 왼쪽으로 옮기자!
- 개발과 동시에 보안성 체크도 함께 이루어지도록…



1.4 Let’s Shift Left!

DEVIEW 2021

Why Shift Left?

- 개발 단계에서 선제적이고 지속적으로 보안 이슈를 발견하고 대응 가능
- 보안과 생산성 사이의 균형을 기대할 수 있음
- 보안 리소스를 보다 deep하고 special한 영역으로 배치할 수 있음
- 개발 과정의 각종 증적을 자동으로 확보해주기 때문에 인증이나 감사 대응이 용이

2. Toothless

2.1 Toothless

DEVIEW 2021

Naver Security가 개발한 DevSecOps System

- DevOps 환경에서 워크플로우가 느려지지 않도록 보안 분석을 자동화
- GitHub enterprise 기반

제공 기능

- 소스코드 정적 분석
- 민감정보 노출 여부 분석
- 오픈소스 취약점 분석
- 웹 어플리케이션 동적 분석
- 보안 이슈 추적/관리

2.2 Approach

이전 기술의 한계

소스코드 정적 분석 외부 솔루션들은 별도의 시스템을 설치/구축/설정 해야하고,
별도의 시스템을 통해 결과를 확인해야 하는 단점이 존재

설계 원칙

1. 네이버의 많은 개발 코드들이 GitHub Enterprise를 이용하여 관리됨
   => GitHub 상에 바로 동작할 수 있는 형태로 연동되어야 한다.
2. 적용 과정이 복잡하면 활용도가 떨어짐
   => Repository에 Toothless를 적용하는 방법이 매우 직관적이고 간단해야 한다.
3. 기존의 CI/CD Pipeline을 해치지 않고 자연스럽게 Integration 되어야 한다.

2.3 Approach – Goal (1/3)

DEVIEW 2021

Continuous/Proactive Security Testing

- 개발 단계에서 선제적이고 지속적으로 보안 이슈를 발견/대응할 수 있어야 한다.

Remove Security Issues Earlier

- 개발 단계에 보안 엔지니어가 투입되지 않아도 개발자가 미리 잡아낼 수 있는 이슈는 미리 잡아내서 수정할 수 있어야 한다.

Security and Development in One Process

- 일반적으로 개발과 보안이 분리되어 별도의 프로세스로 존재하는데, Shift Left를 통해 자연스럽게 보안과 개발을 연결시킬 수 있어야 한다.

2.3 Approach – Goal (2/3)

Balance Between Security and Productivity

- Shift Left를 통해 개발 단계에서 보안 체크가 이루어지게 함으로써 빠른 개발과 배포 주기를 가져야 하고 보안과 생산성 사이의 균형을 기대할 수 있어야 한다.

Security Engineer Resource

- Shift Left가 되면 보안 리소스를 보다 deep하고 special한 영역으로 배치할 수 있는 장점이 있으며, 개발자가 해결하지 못하는 보안 취약점 이슈나 보안 모니터링, 모의 해킹 등에 좀더 초점을 맞출 수 있게 해야 한다.

Compliance

- 보안 인증이나 감사, 실태 조사 등을 대응하기 위해서는 개발 과정의 각종 증적이 필요한데 이를 위한 증적을 자동으로 확보해줄 수 있어야 한다.

2.3 Approach – Goal (3/3)

DEVIEW 2021

Seamless & Automation

- 기존의 CI/CD pipeline에 자연스럽게 통합되어 개발 과정에서 자동으로 동작해야 한다.

Simple & Clear & Actionable

- 복잡하지 않고 단순하면서 보안 체크 결과가 명확해서 개발자가 결과를 보고 어떤 대응을 해야 하는지 판단할 수 있어야 한다.

Developer Friendly

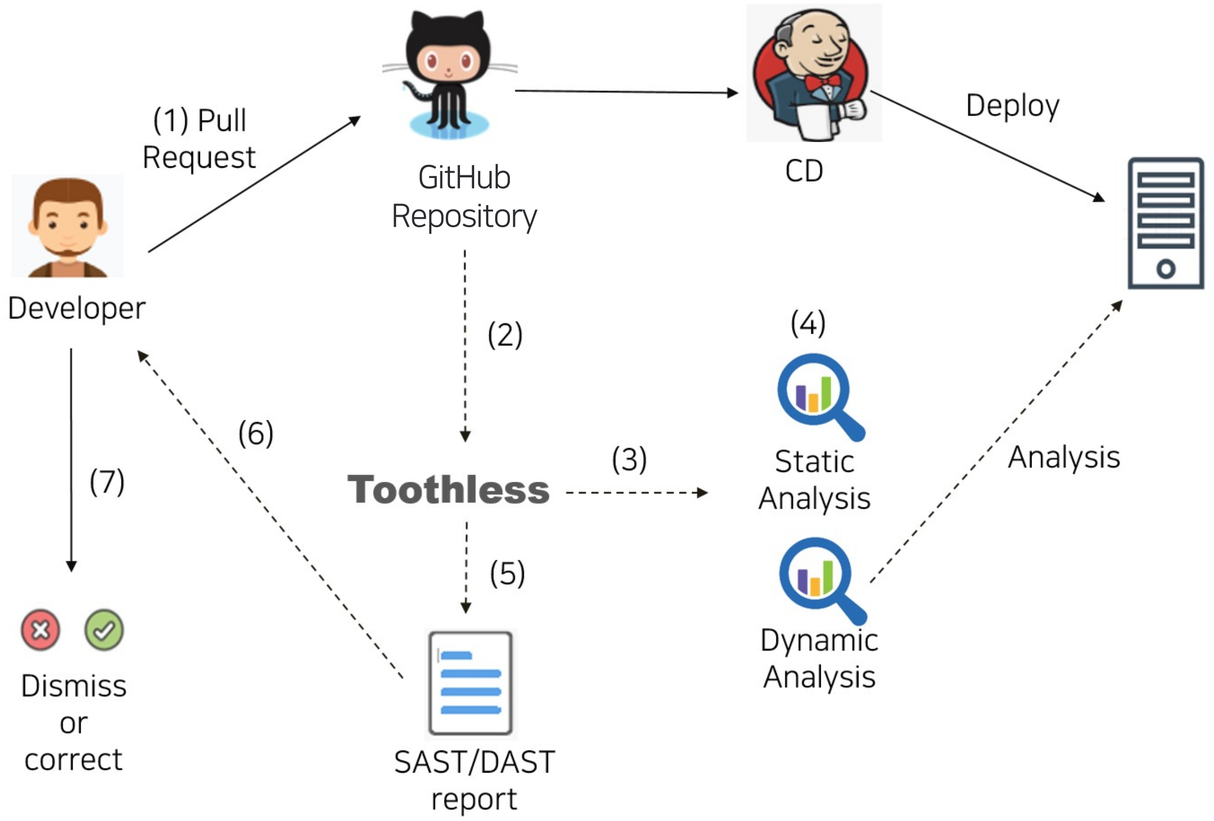
- 개발자가 느끼는 거부감을 최소화하는 방식이 되어야 한다.

Multi-Language

- 다양한 개발 언어를 지원해야 한다.

2.4 Action Flow

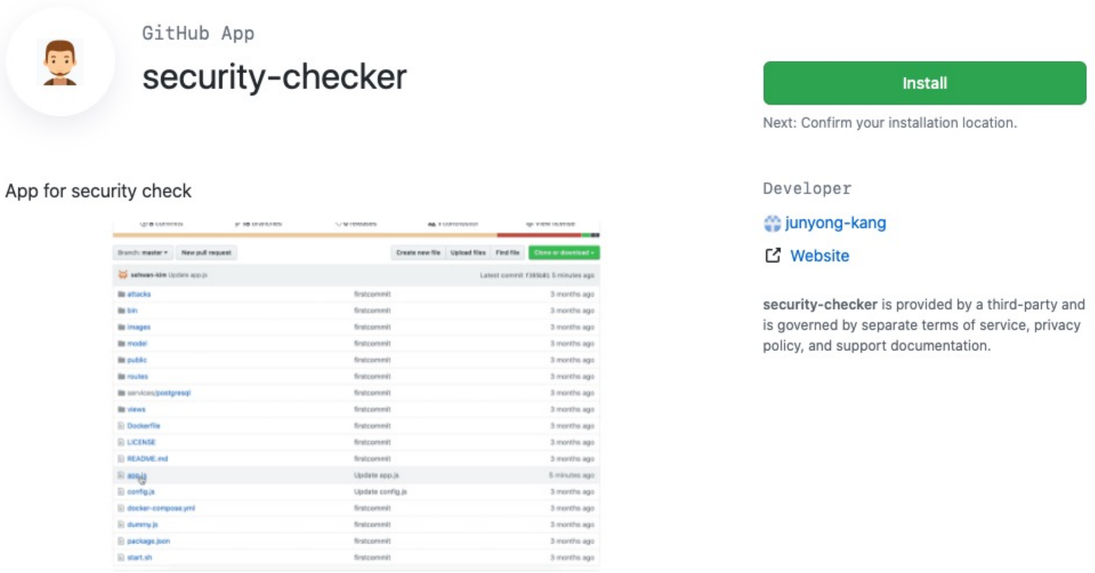
Toothless의 모든 동작은 GitHub과 연계되어 자동으로 이루어짐
개발자가 Pull Request를 요청하는 시점에 repository에 있는 소스코드에 대한 취약점 분석을 시작
분석 결과도 GitHub의 내부 페이지로 제공

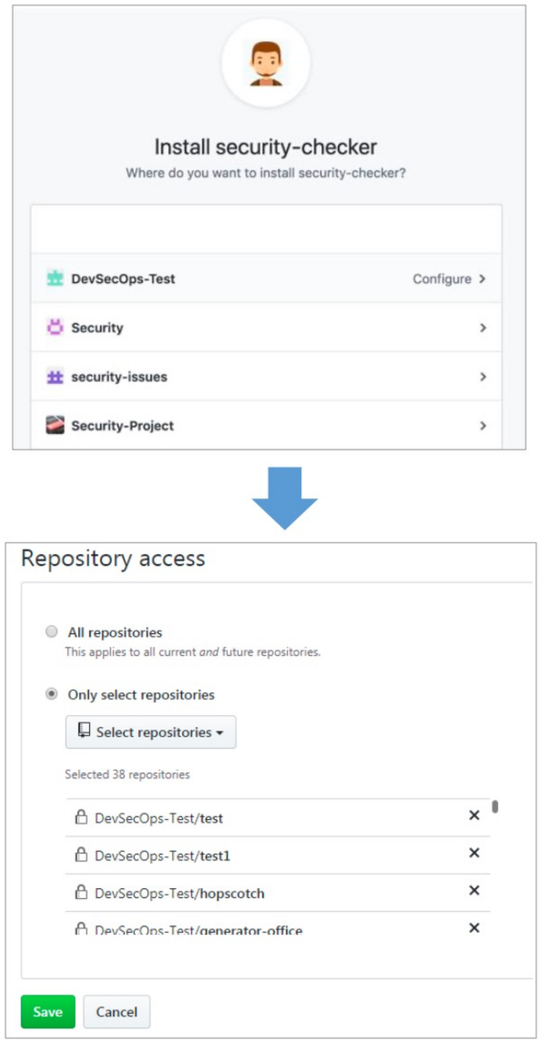


2.5 Easy To Setup

적용과 사용이 매우 간단함

- GitHub App 설치 페이지에서 클릭 몇 번만으로 가능



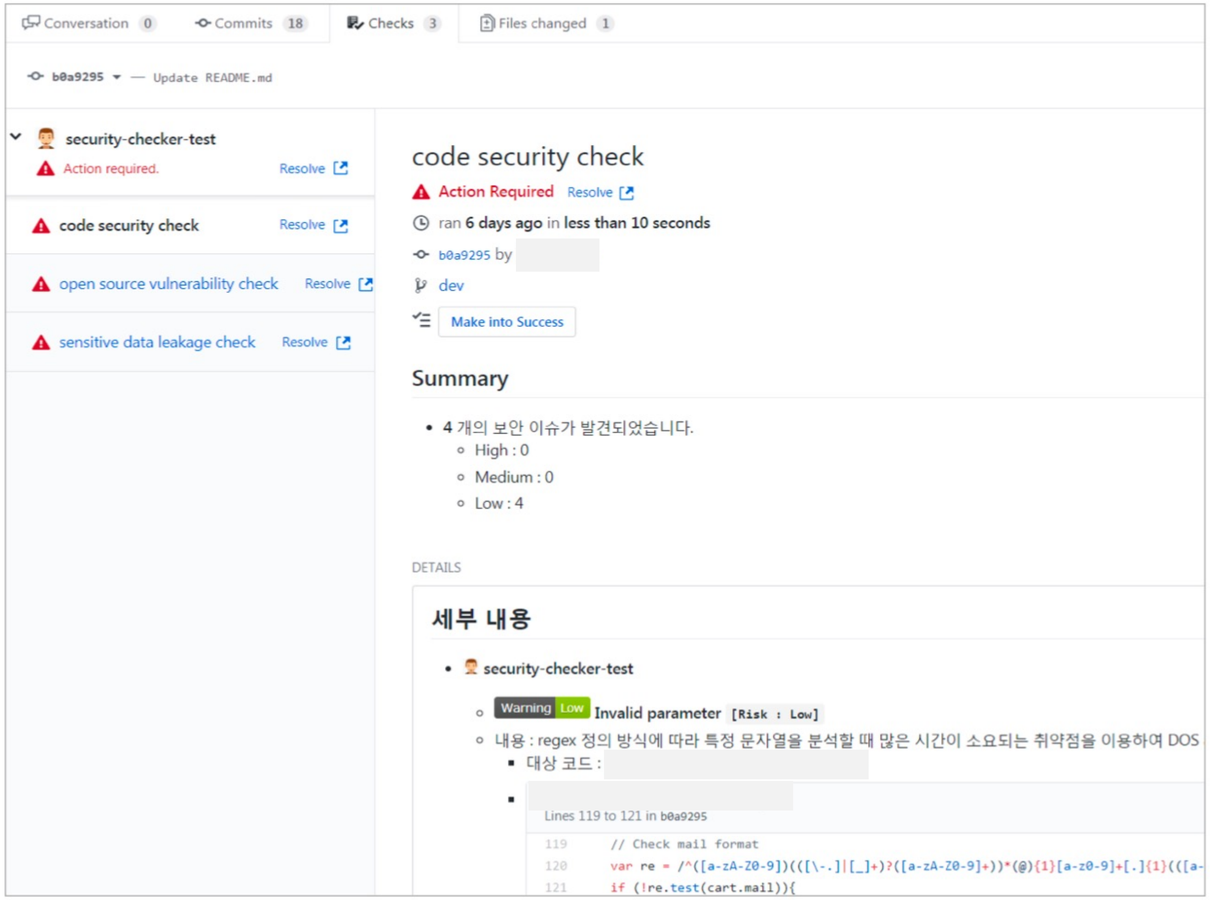


3. 주요 / 부가 기능

3.1 소스코드 정적 분석

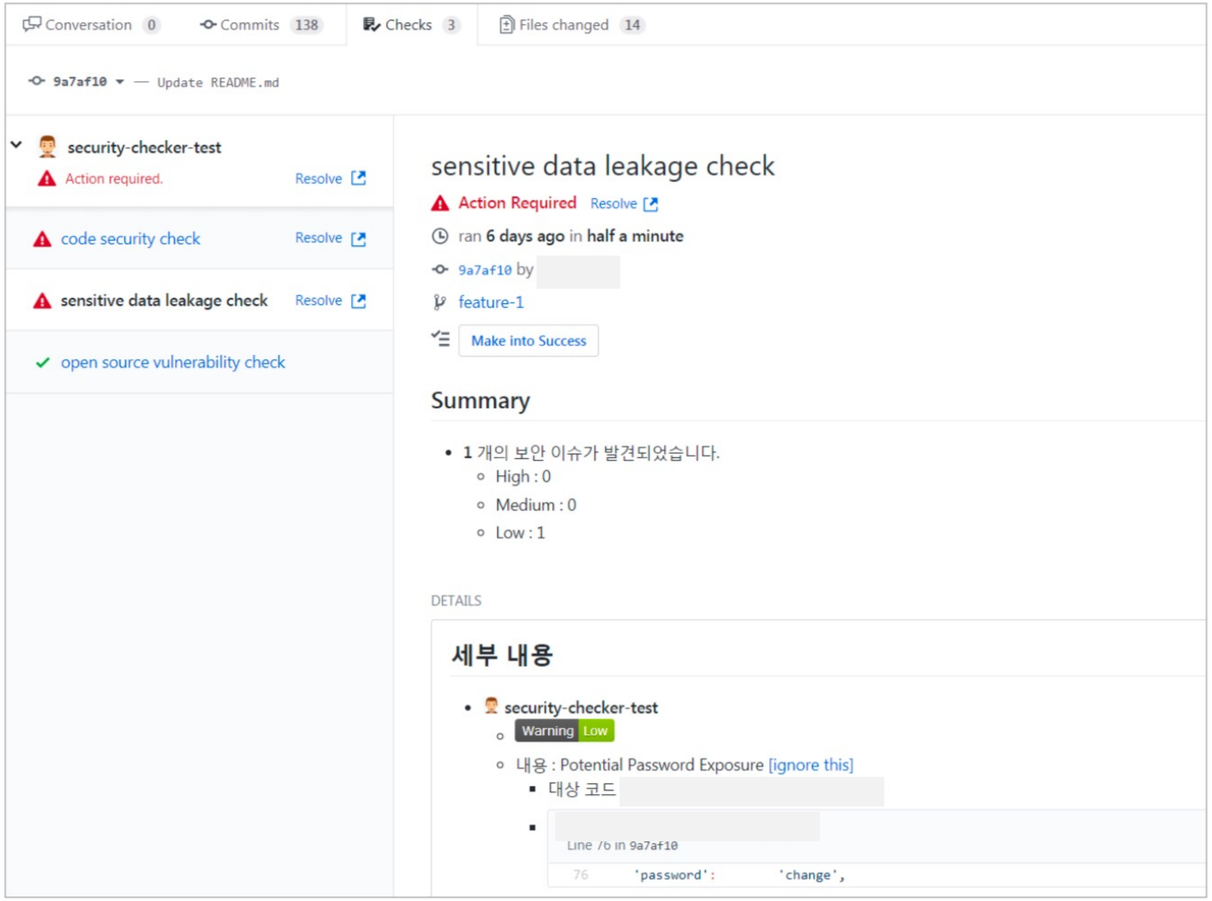
DEVIEW 2021

소스코드를 분석하여 코드 상에 존재하는 보안 취약점을 탐지



3.2 민감정보 노출 여부 분석

패스워드나 토큰 정보 등 Hardcoding 되어 있는 민감 정보를 스캔

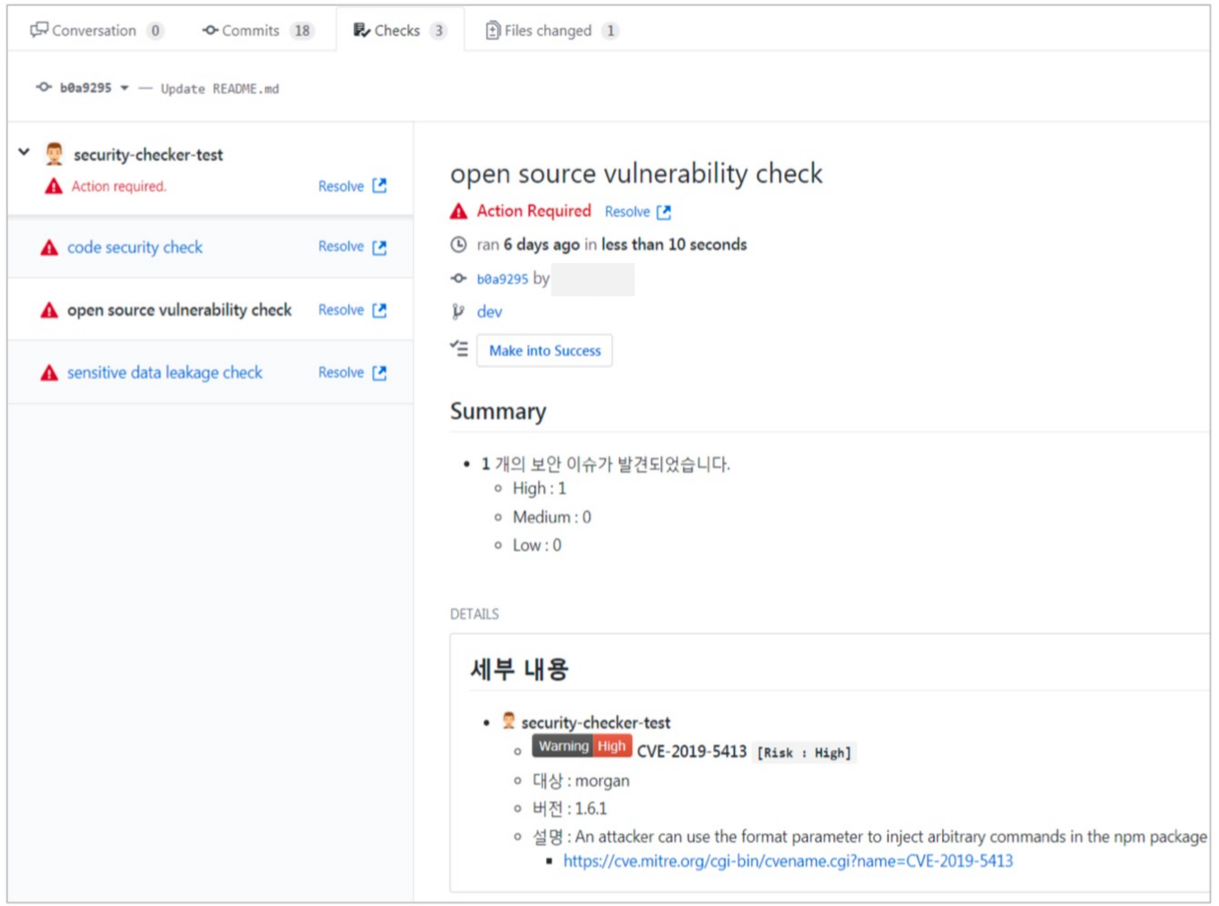


3.3 오픈소스 취약점 분석

DEVIEW 2021

개발 프로젝트에서 사용하고 있는 오픈소스를 탐지하여 취약점을 리포팅

사용 중인 오픈소스 라이브러리
버전에 대한 CVE를 확인한 후,
취약점 존재 여부를 판단



3.3 오픈소스 취약점 분석

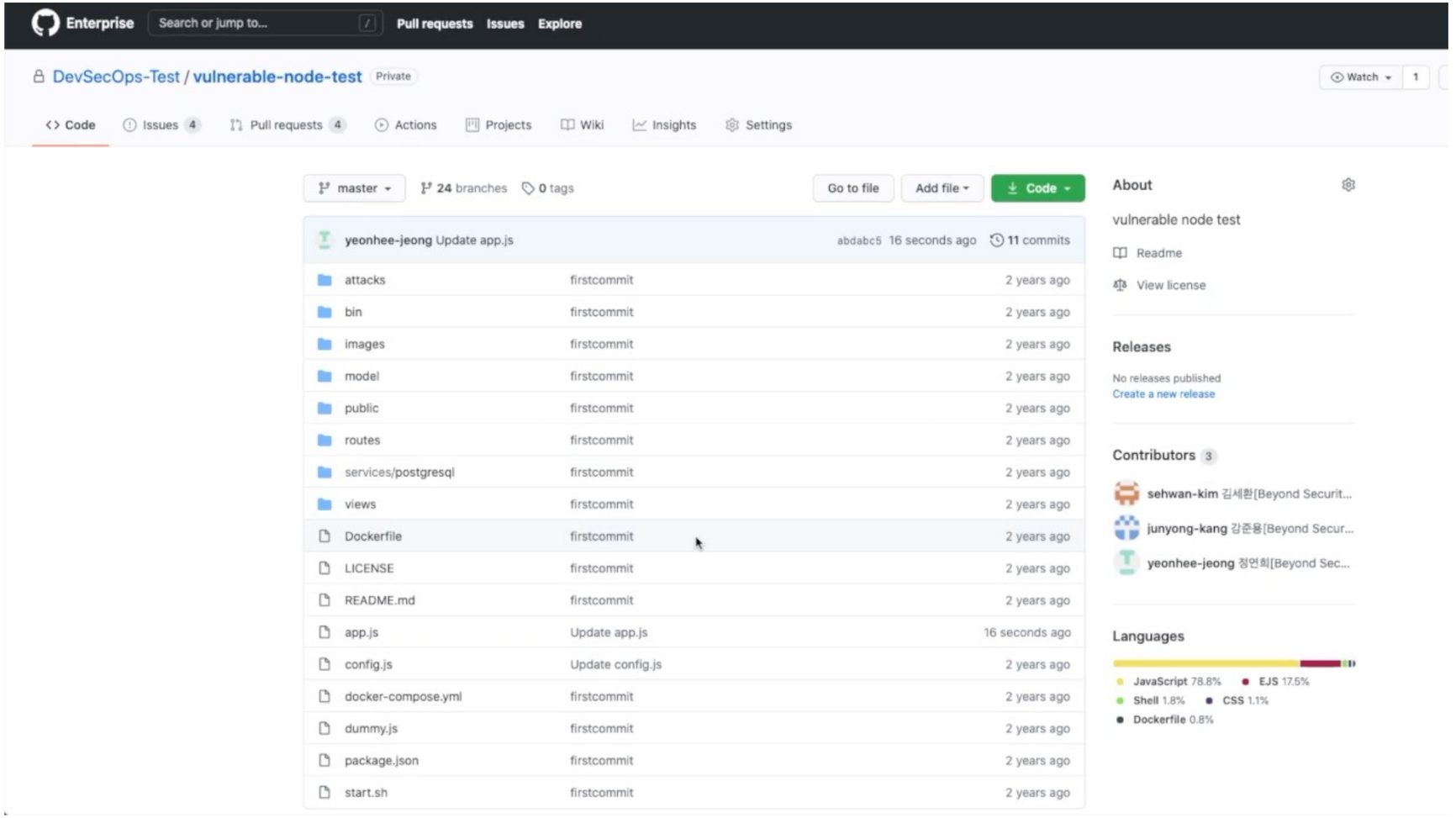
Open Source 취약점 검사 필요 파일

각 언어별로 패키지 매니저 관련 파일이 repository 내에 존재해야 분석 가능

언어	패키지 매니저	필요 파일
Java	Maven	pom.xml
Java	Gradle	build.gradle
Javascript	NPM (Yarn)	package.json
Python	PIP	requirements.txt
Go	Dep	gopkg.lock, gopkg.toml
Go	Modules	go.mod, go.sum
C (C++)	CMake	cmakelists.txt
Ruby	Bundler	gemfile
Objective C (Swift)	CocoaPods	podfile.lock

* Demo

DEVIEW 2021

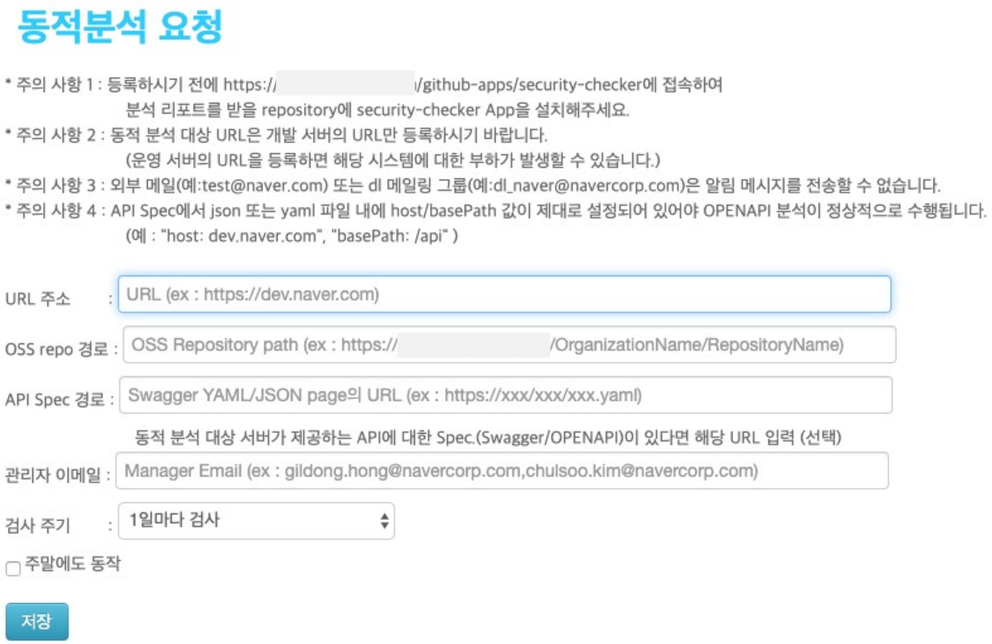


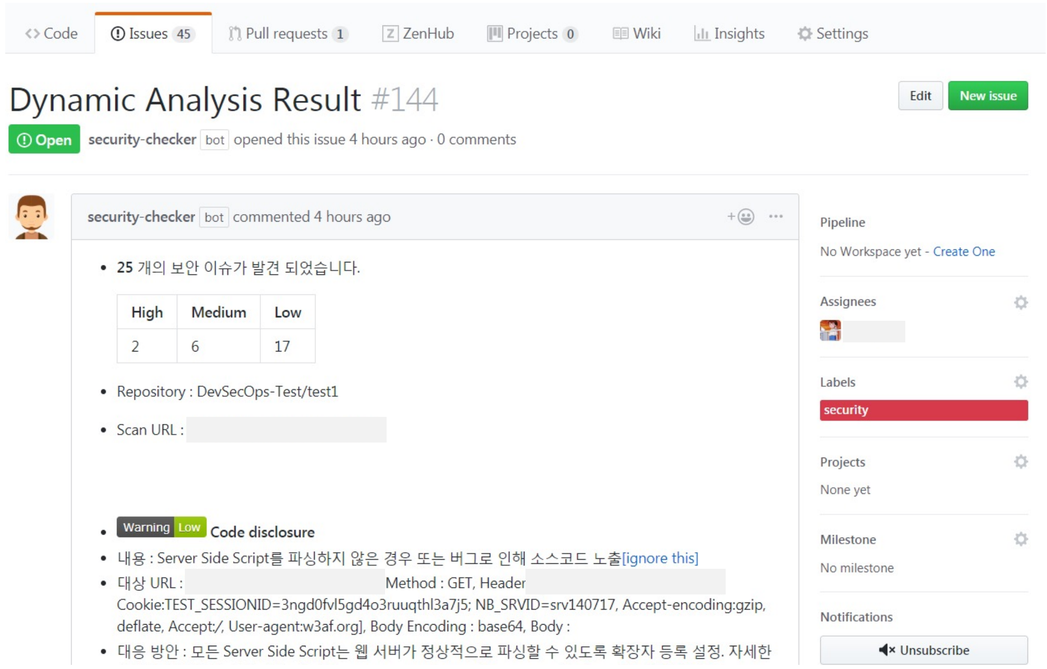
3.4 웹 어플리케이션 동적 분석

DEVIEW 2021

개발 서버에 배포되어 동작하고 있는 웹 어플리케이션에 대한 보안 체크

- 결과를 해당 Git repository의 Issue로 자동 레포팅





3.5 부가 기능 (1/3)

DEVIEW 2021

Customize Settings

개발자가 자신이 원하는 대로 분석 설정 가능

- 리포팅을 받을 취약점 레벨 설정
- 분석에서 제외할 파일/디렉토리 설정
- 분석에서 제외할 특정 탐지 룰 설정
- 특정 확장자 파일만 분석하도록 설정

```
{
    // High & Medium & Low  : 3
    // High & Medium        : 2
    // High                 : 1
    // 예) "ReportLevel "  : 1

    // 검사에서 제외할 폴더 리스트(string)
    // 내용 : full path에서 repository 이름 이후의 폴더 경로를 입력
    // 예) "ExcludeDir" : ["src/test", "src/test2"]

    // 검사에서 제외할 파일 리스트(string)
    // 내용 : 파일 명을 입력
    // 예) "ExcludeFile" : ["test_file.c", "test_file2.c"]

    // 검사에서 제외할 탐지 룰 리스트(string)
    // 내용 : 탐지 룰 이름을 입력
    // 예) "ExcludeCheckRule" : ["Invalid parameter", "Information exposure"]

     // 검사를 수행할 파일 확장자 리스트(string)
     // hrj4a : 파일 확장자 입력
     // 예) "ScanFileExtension" : [".java", ".c"]

}
```

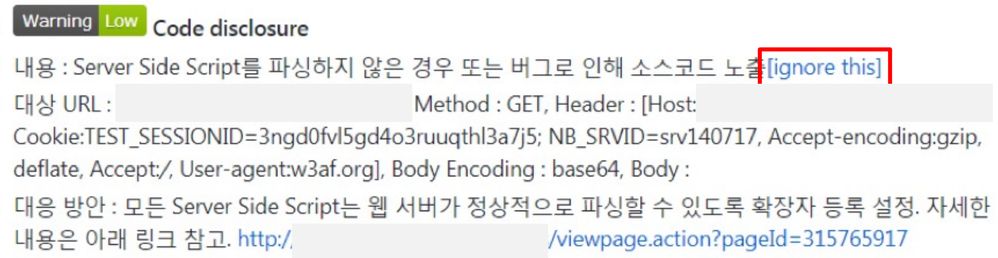
3.5 부가 기능 (2/3)

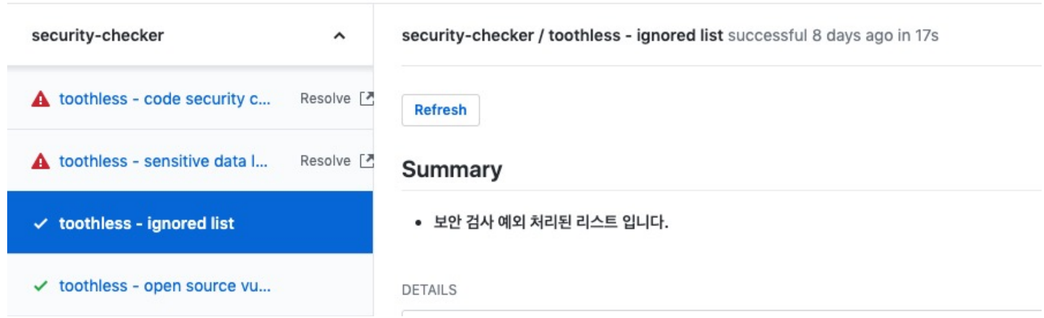
DEVIEW 2021

Ignore Warnings

리포팅된 보안 취약점이 known issue거나
오탐인 경우 분석 리포트에서 제외 가능

- Ignore 처리한 이슈 목록을 확인할 수 있으며,
  다시 ignore 해제하는 것도 가능

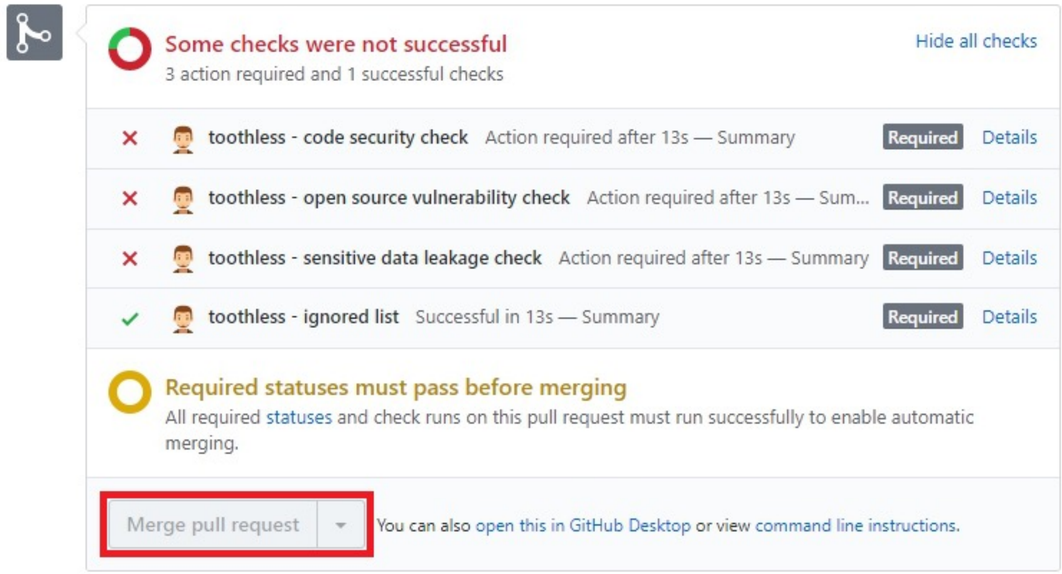




3.5 부가 기능 (3/3)

Merge Protection

Toothless의 분석 결과 중 통과하지 못한
분석 항목이 있다면 Merge가 되지 않도록 설정



3.6 Toothless-Wing (1/2)

DEVIEW 2021

Toothless의 Issue Tracking System

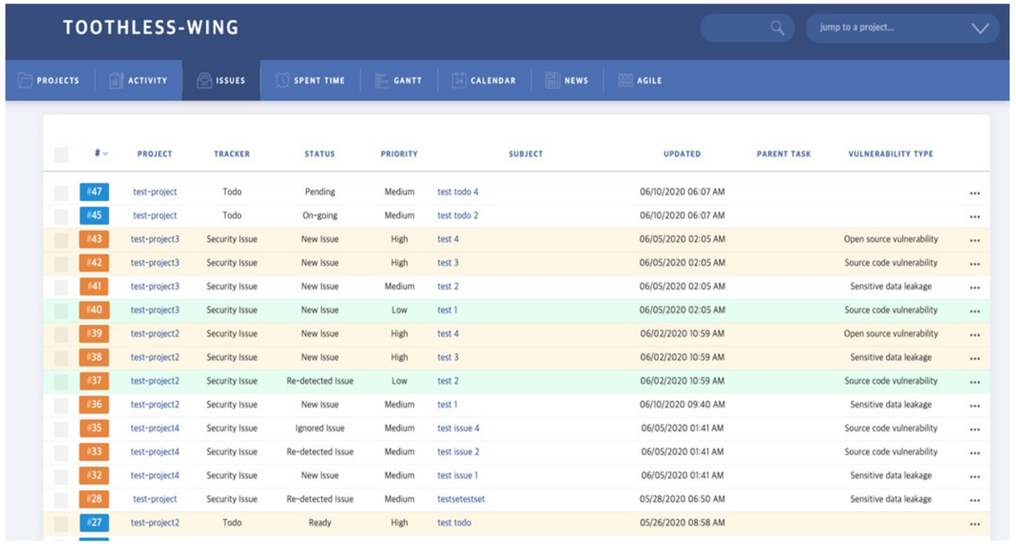
- Toothless에서 발견된 이슈를 자동으로 등록 및 관리
- 일정관리/통계 등과 같은 다양한 부가기능 제공

3.6 Toothless-Wing (2/2)

DEVIEW 2021

Toothless의 Issue Tracking System

- 발견된 보안 이슈 목록과 각 보안 이슈의 내용/상태를 확인 가능
- 각 보안 이슈를 해결하기 위한 액션(To-Do)을 등록하여 액션 플랜을 수립할 수 있음



4. 개발 과정

4.1 사전 인터뷰 수행

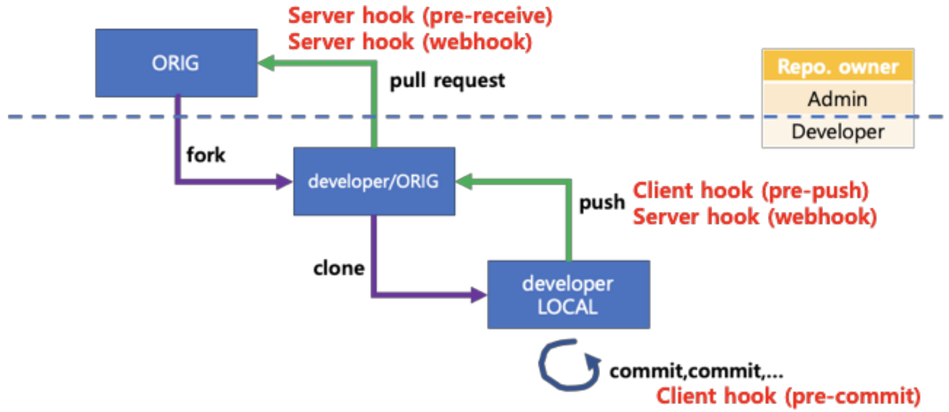
네이버의 각 서비스 별 현황 파악

- 서비스 구성, 주 사용 언어, 현재의 CI/CD Flow
- PR/Merge 정책, 배포 주기
- Secure Coding 고려 여부, 소스코드 정적 분석 도구 적용 여부
- 오픈소스 관련 점검 여부, 보안 검수 여부
- 개발자 추가 의견
  - DevSecOps 적용에 대해 방어적으로 생각하는 포인트
  - DevSecOps가 지원해주면 좋을 것 같은 포인트

4.2 GitHub 연동

GitHub 활용의 기본 flow를 기반으로 리서치

- 사용 가능한 Hook의 종류
- Branch API 종류



소스코드 분석 후 결과를 보여줄 공간 확보

- 특정 commit에 comment 추가, Issue 탭 이슈 생성, 특정 PR에 review 추가, …
- Check 탭에 레포트 생성 -> GitHub App을 통해서 가능 (인증 문제)

4.3 분석 툴 확보

개발 언어별 SAST/DAST 툴 리서치

- 오픈소스 Repository에 직접 적용해서 테스트
  - 분석 툴 별로 command line option, report 포맷 확인
  - 오탐률, 분석결과 비교
- 입맛에 딱 맞는 경우는 잘 없었으며, 많은 customizing 작업이 동반됨

적당한 툴이 없는 경우 자체적으로 구현

- Android App Code Scanner
- JavaScript Framework Code Scanner

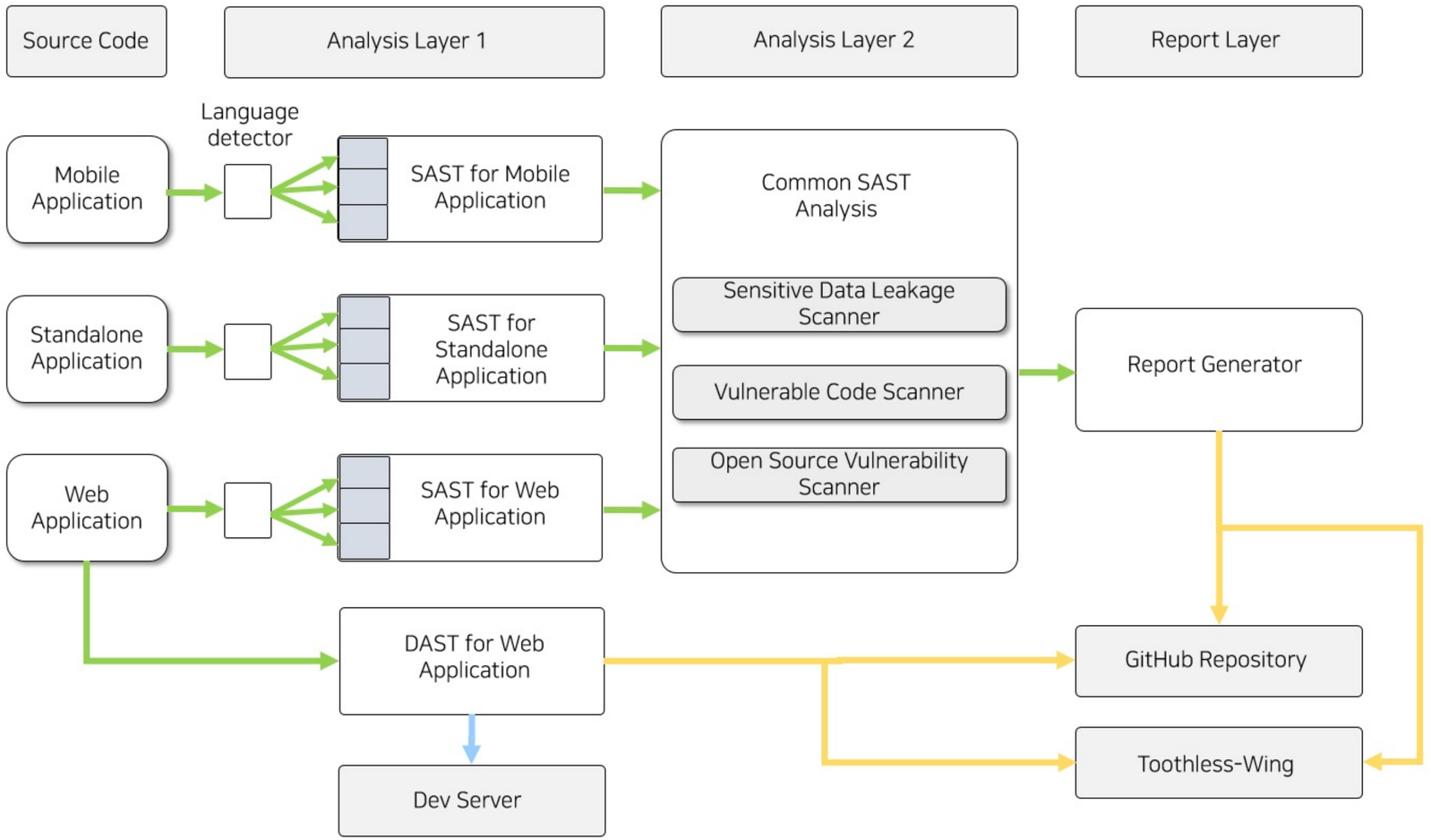
4.4 레포트 생성

개별 분석 툴마다 지원하는 레포트 포맷이 다른 문제

- 각 레포트 포맷을 파싱해서 원하는 정보를 추출해서 취합, 동일 포맷으로 출력

	HTML	TEXT	JSON	CONSOLE	Stylish	prose	EMACS	Mark Down	…
Tool A	O(default)	X	X	X	X	X	X	X	…
Tool B	X	O(default)	X	X	X	X	X	X	…
Tool C	O(default)	X	X	X	X	X	O	X	…
Tool D	O	X	O	X	O(default)	X	X	X	…
Tool E	X	X	O	X	O	O(default)	X	X	…
Tool F	X	X	O(default)	X	X	X	X	X	…
Tool G	O	O	O	O(default)	X	X	X	X	…
Tool H	X	X	X	O(default)	X	X	X	X	…

4.5 Analysis Flow



4.6 Trouble Shooting (1/4)

개발자 친화적으로 만들기 (개발자의 거부감을 최소화 하기)

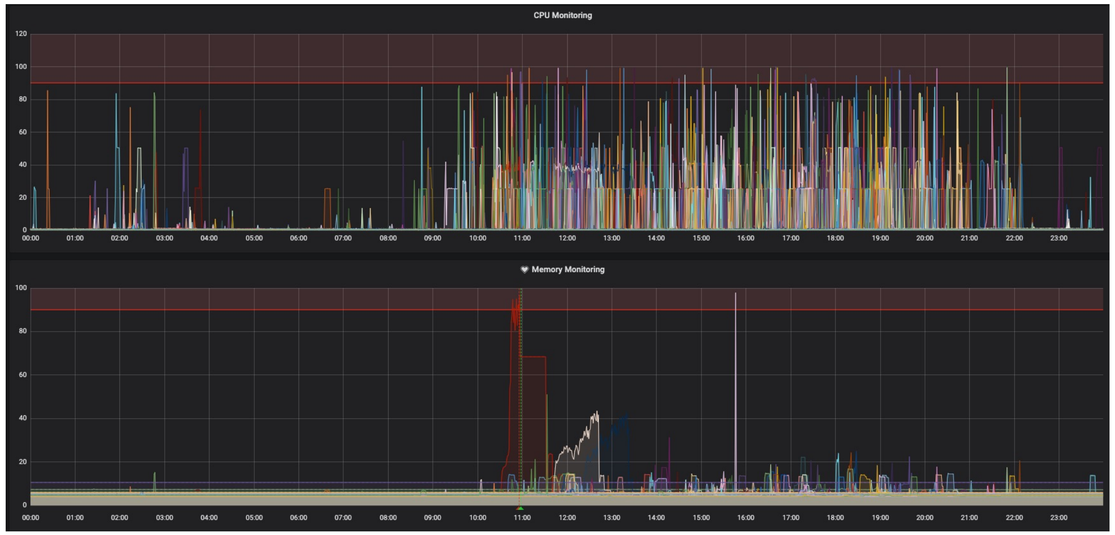
- 가장 주요한 목표 중 하나이자 동시에 가장 어려웠던 점
- 설계 단계 이전부터 사전 인터뷰를 통해 의견 청취
- 베타 테스트, 정식 오픈 이후에도
  주기적으로 사용성 설문조사
- 직관적이고 친절한 사용 가이드 제공
- 질문이나 제안사항을 쉽게 등록할 수 있는 창구 마련
- 제안사항을 통해 Next item 도출

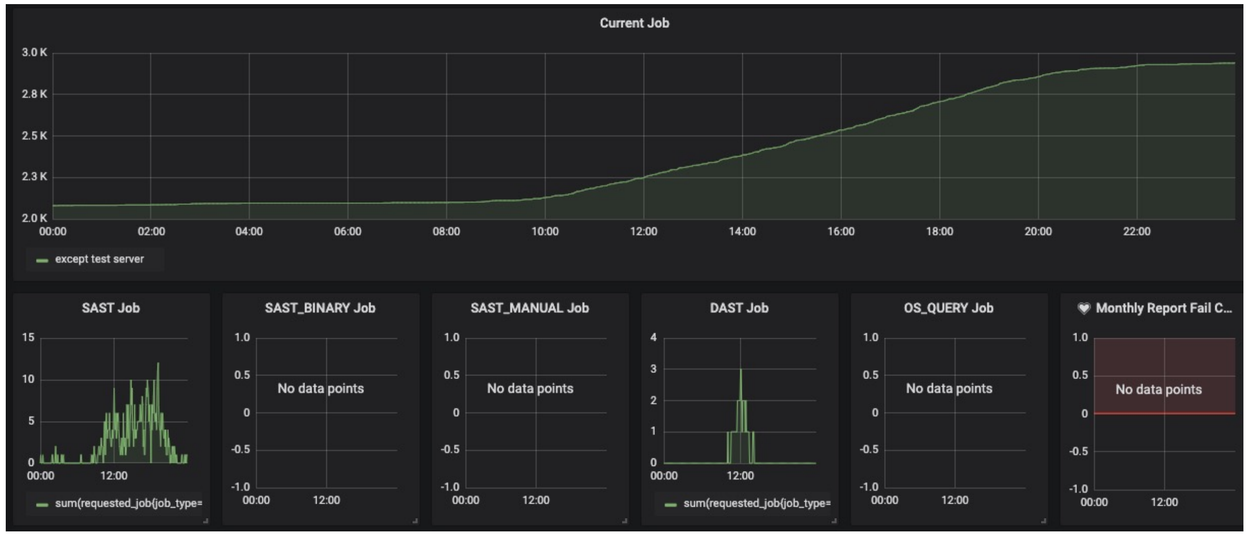
4.6 Trouble Shooting (2/4)

DEVIEW 2021

너무나 다양한 소스코드들 => 예상하지 못한 에러 발생

- 매일 하루도 빠지지 않고 자체 모니터링 수행
- CPU/Memory/Disk 모니터링, 분석 서버 Health check, Job count
- 특이사항, 에러 발생 시 해당 repository 담당자에게 접근권한 요청하여 직접 테스트



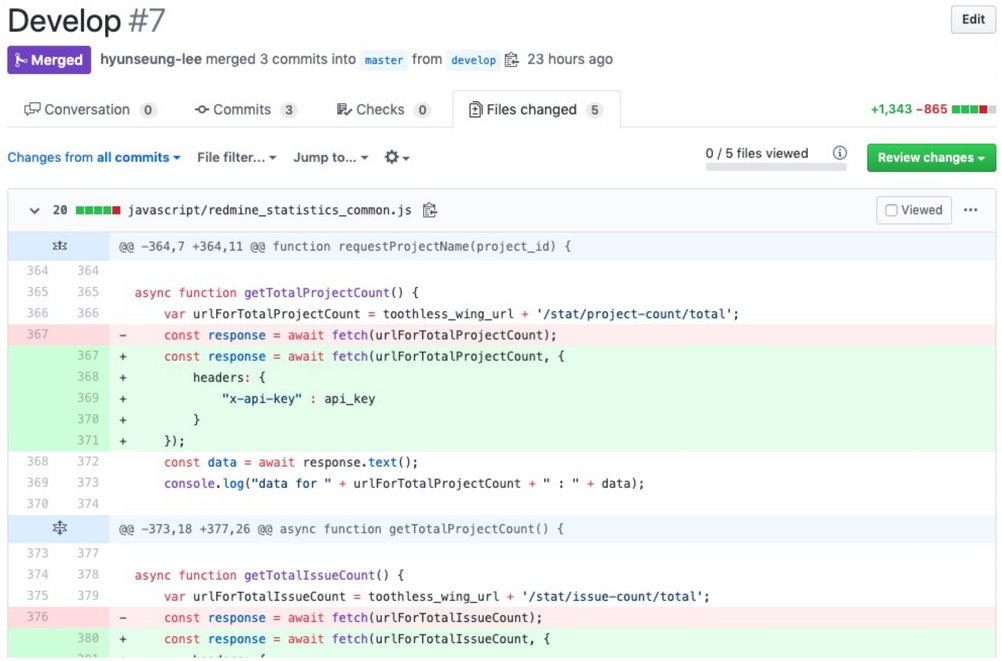


4.6 Trouble Shooting (3/4)

DEVIEW 2021

분석 속도

- 일반적으로 용량이 작은 repository의 경우 분석 결과가 빠르게 도출되지만, 100M 이상의 대용량 repository의 경우는 다소 오랜 시간이 소요됨
- 분석 도구 마다 분석 속도가 상이함 (일률적인 분석 속도를 얻기 어려움)
- Differential analysis 기능을 적용
  - 기존 코드 대비 수정된 부분만 분석



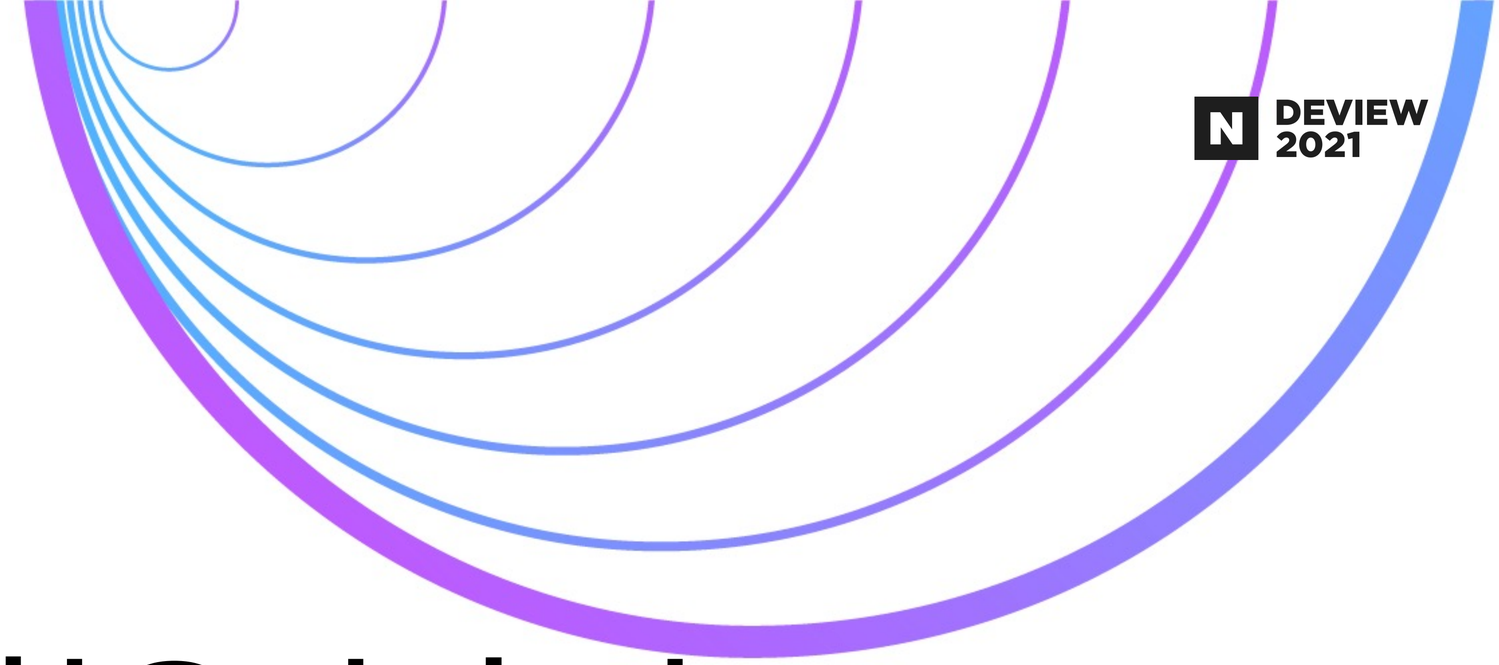
4.6 Trouble Shooting (4/4)

다양한 개발 언어 지원

- 각 언어별로 보안이 취약한 개발 패턴에 대한 리서치가 필요했음
- 사실상 지름길은 없고, 많은 시간을 투자해야 했음

탐지 룰 가다듬기

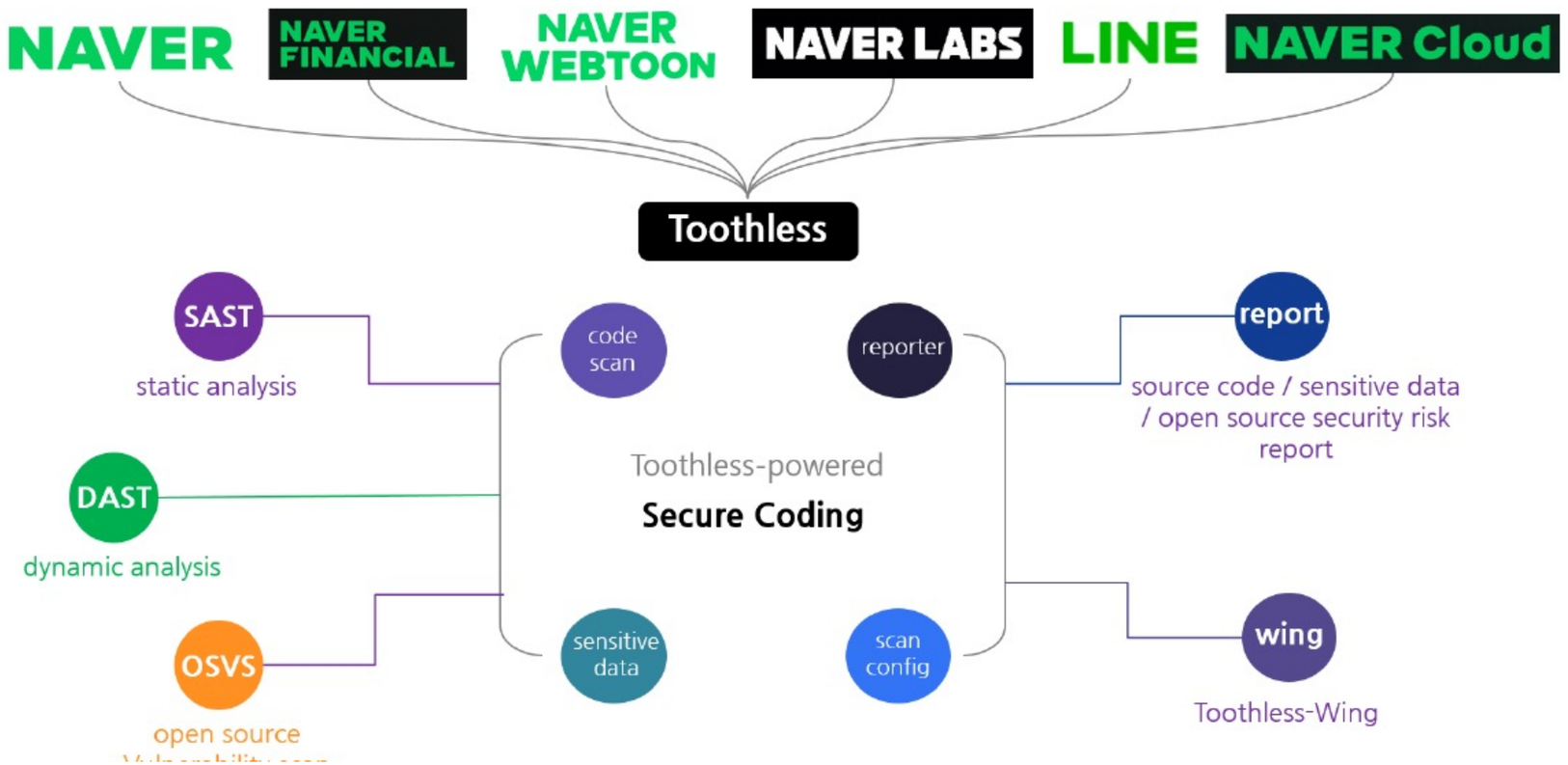
- Toothless의 보안 탐지 룰이 400개가 넘으며,
  각각의 룰에 대해 타당성을 확보하고 위험도를 분류함
- Security 내부 구성원들의 리뷰 및 취약점 전문가들의 의견 검토 과정이 들어감



5. 활용과 효과

5.1 적용 현황 (1/2)

2020년 6월 정식 오픈 이후 전 계열사 적용



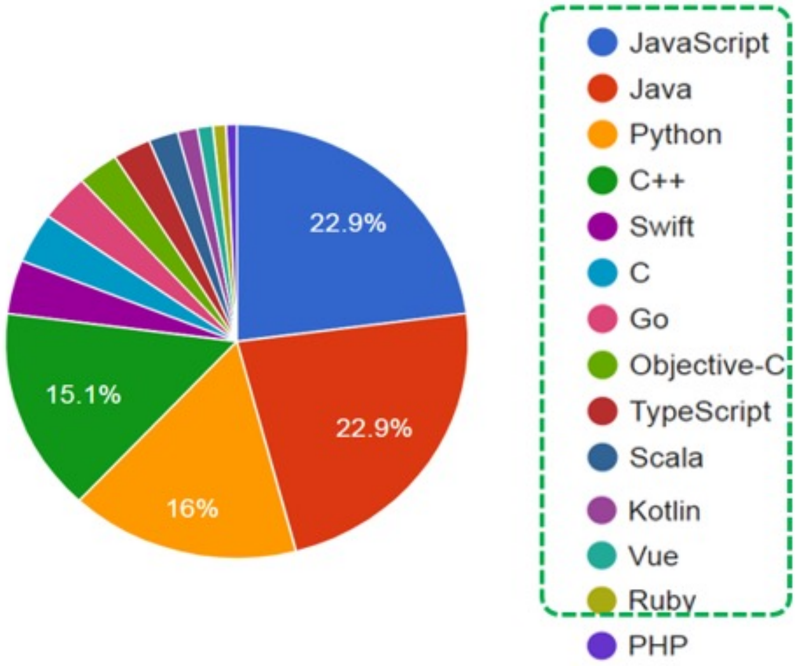
5.1 적용 현황 (2/2)

DEVIEW 2021

매일 1100회 이상의 분석을 수행 중

5.1 적용 현황 – 지원 언어

네이버에서 주로 사용하는 개발 언어 및 Toothless 지원 범위



언어	분석 기능
Java	소스코드 취약점 검사, 오픈소스 취약점 검사, 민감정보 노출 검사
Javascript/Typescript	소스코드 취약점 검사, 오픈소스 취약점 검사, 민감정보 노출 검사
Go	소스코드 취약점 검사, 오픈소스 취약점 검사, 민감정보 노출 검사
C (C++)	소스코드 취약점 검사, 오픈소스 취약점 검사, 민감정보 노출 검사
Kotlin	소스코드 취약점 검사, 오픈소스 취약점 검사, 민감정보 노출 검사
Python	오픈소스 취약점 검사, 민감정보 노출 검사
Ruby	오픈소스 취약점 검사, 민감정보 노출 검사
Objective C (Swift)	오픈소스 취약점 검사, 민감정보 노출 검사
Scala	소스코드 취약점 검사, 민감정보 노출 검사

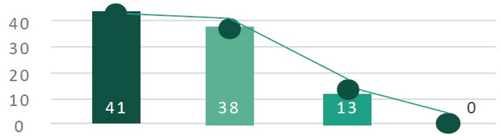
5.2 보안성 개선 사례

실제 서비스 부서 Repository의 검출 Issue 수 추이

source code
security vulnerability

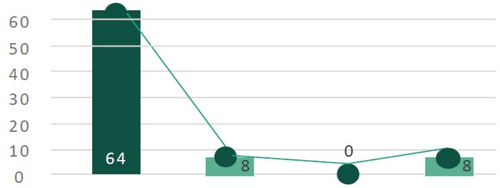
- Organization : S**
- Repository : S*******

☞ 100% 개선



- Organization : W********
- Repository : b*******

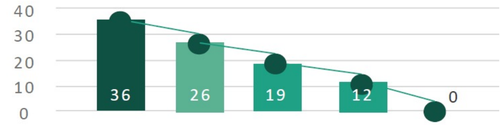
☞ 87.5% 개선



Sensitive Data Leak

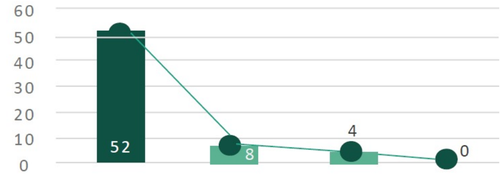
- Organization : N****
- Repository : S**********

☞ 100% 개선



- Organization : N********
- Repository : b**********

☞ 100% 개선



5.3 지속적인 업데이트

- Repository내에 있는 인증서 만료 기간 체크
- 개인 정보 DB에 접근하는 소스 코드 여부 판단
- GraphQL introspection을 통한 API scheme 노출 체크
- 정적 분석 지원 언어 추가
- 다국어 지원
- Toothless badge 기능 추가 

- 분석 툴 업데이트 및 정확도 개선
- ...ing

6. Toothless의 미래

6.1 Next Slogan

DEVIEW 2021

“ Developer-First Security “

- 기존의 Shift Left에서 한발 더 나아가서, 좀 더 강화된 개발 단계에서의
  보안 체크와 개발자 스스로 주도적으로 할 수 있는 보안 체크를 지원하자!

- 기존: 개발이 완료된 소스 코드를 대상, PR 수행 시에 보안 체크
- Developer-First Security: 코드를 작성하는 중에도 보안 체크가 이루어지도록
- 개발 프로세스의 전반적인 과정에서 보안 체크가 이루어지도록 확장

6.2 Developer-First Security

DEVIEW 2021

Developer-led Security Check

- 개발자 주도로 보안 체크가 이루어지도록 만들자.

Security Check in the Overall Development Process Step

- 보안 체크는 개발 프로세스의 전반적인 과정에서 이루어지도록 만들자.

Developer Friendly

- 이를 위해서는 (역시)개발자 친화적인 시스템이 되어야 한다.

Preventing Security Bugs by Developer

- 결국, 개발자가 보안 버그를 만들지 않도록 방지하는 것이 목적이다.

6.3 Next Items

DEVIEW 2021

Developer-First Security 컨셉에 맞게 고려 중인 Item들

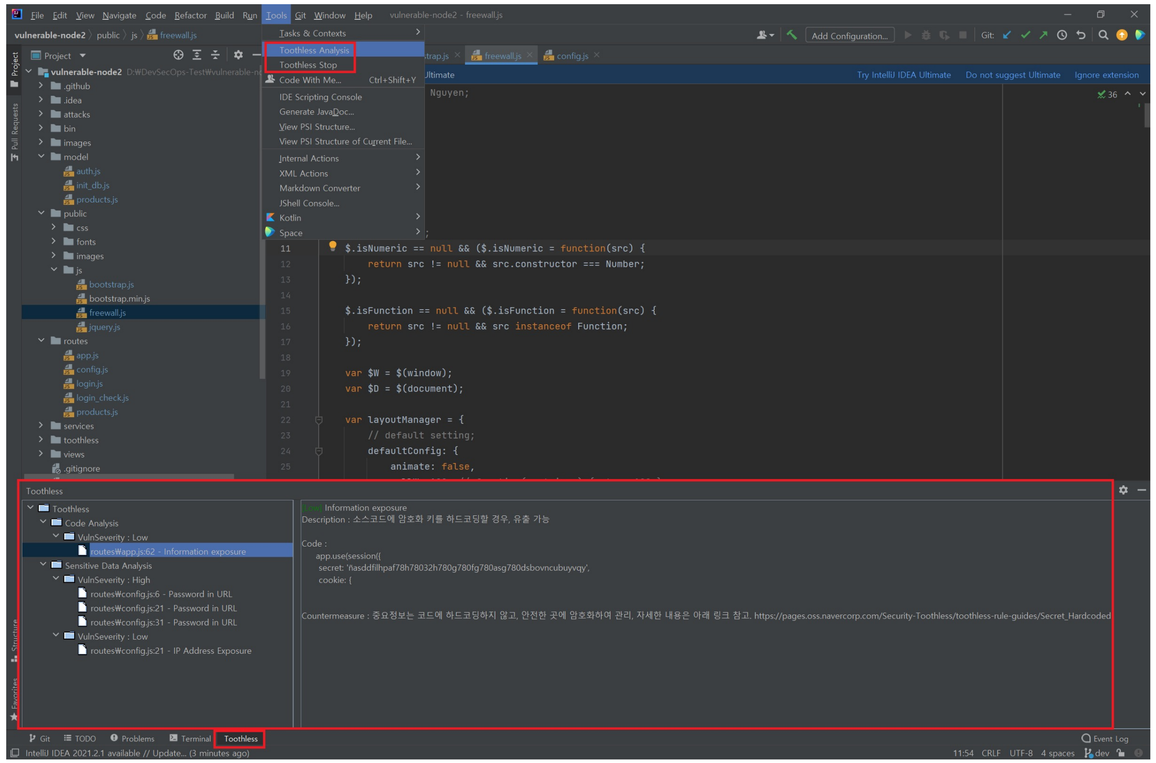
- 개발자가 IDE를 이용하면서 개발을 함과 동시에 스스로 보안 체크를 할 수 있는 IDE Plugin 지원
- 개발자가 스스로 소스 코드를 체크할 수 있는 Rule을 만들어서 적용할 수 있는 기능 지원
- 개발자가 스스로 만든 소스 코드 체크 Rule을 서로 공유하거나 관련 정보를 서로 공유하고 토론할 수 있는 community platform 제공
- 개발자 간에 공유되는 소스 코드 체크 Rule을 IDE Plugin에 바로 적용할 수 있는 기능 제공

6.4 New Features (1/2)

IDE Plugin

개발자가 자신의 local IDE에서 간단히 작업 중인 소스코드를 검사할 수 있도록 지원

- 가장 많이 사용되는 IDE 중 하나인 IntelliJ에 대한 Plugin을 기반으로 점차 다양한 IDE를 지원할 예정
- 현재 베타테스트 단계

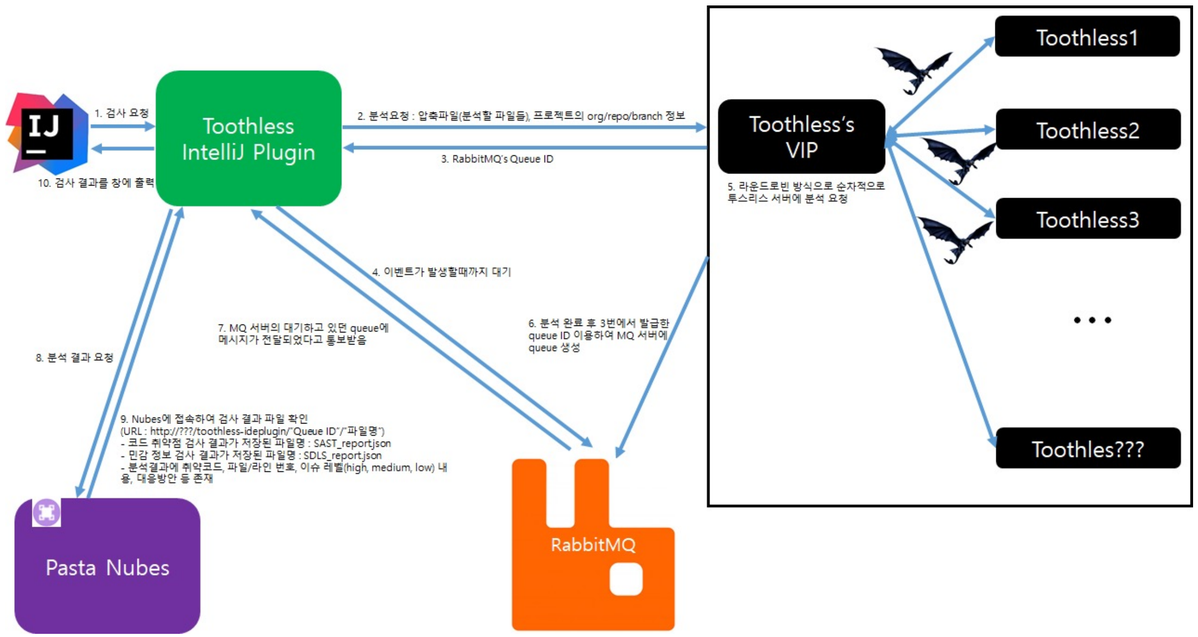


6.4 New Features (1/2)

DEVIEW 2021

IDE Plugin

개발자가 자신의 local IDE에서 간단히 작업 중인 소스코드를 검사할 수 있도록 지원



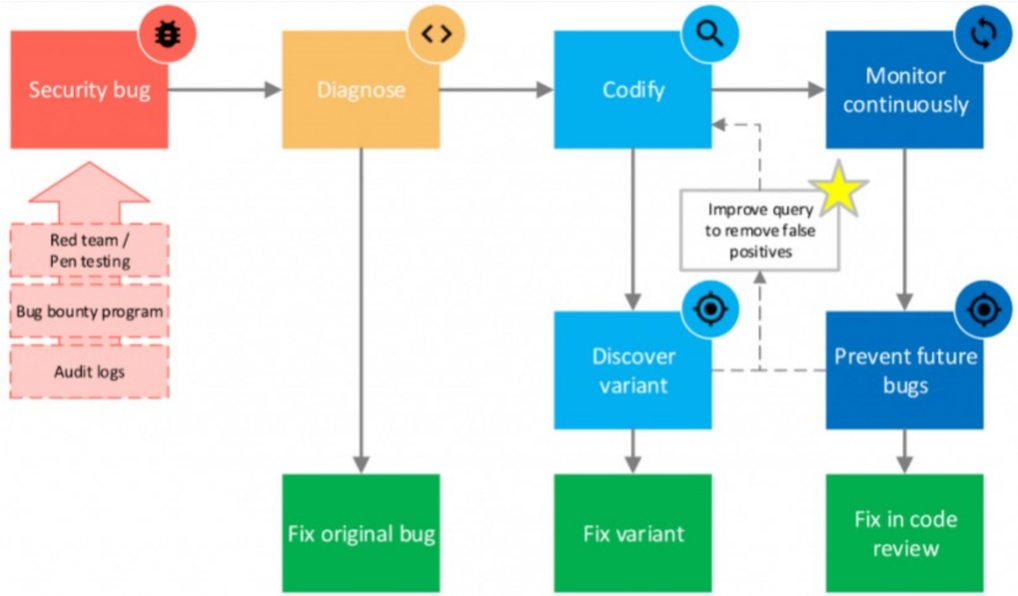
6.4 New Features (2/2)

DEVIEW 2021

Variant Analysis

유사 취약점 탐지를 위한 variant analysis 엔진 개발

- 신규 취약점의 코드 특징을 분석해 패턴화
- 해당 패턴을 등록하여 유사한 코드를 검출
- 발견된 취약점에 대한 내용 뿐만 아니라 개발자가 자체적으로 고안해 낸 Rule을 적용하는 것도 가능하게 함



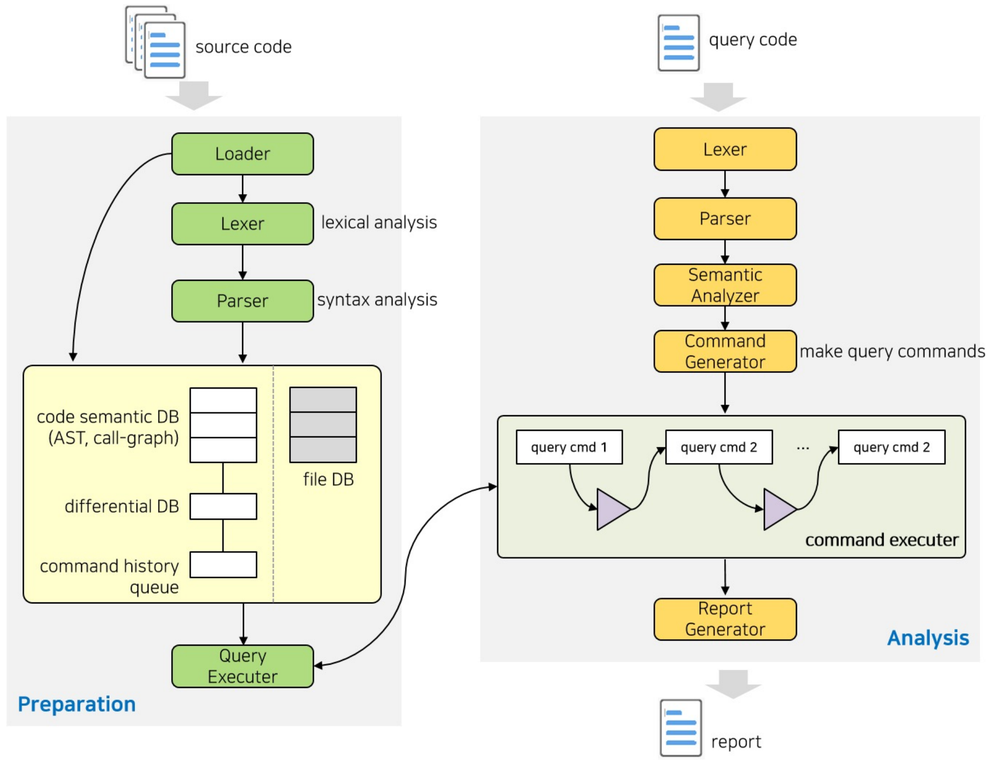
https://devblogs.microsoft.com/devops/microsoft-devsecops-static-application-security-testing-sast-exercise

6.4 New Features (2/2)

Variant Analysis

유사 취약점 탐지를 위한 variant analysis 엔진 개발

- 현재 Java 언어를 대상으로 개발
  (추후 JavaScript 등 순차적으로 언어 확장)
- 소스코드 파싱(Lexer, Parser)을 직접 구현
- 사용하기 쉬운 Query 언어 설계



Thank You