

STATIC **Enterprise / Standalone**

코드 정적 검증 도구

SURESOFT

TABLE OF CONTENTS

Why STATIC	3p
Needs	4p
Solution	5p
Specification	21p
References	22p

The background is a solid dark blue. It features several decorative elements: a series of thin, light blue wavy lines that sweep across the lower half of the image, and three solid blue circles of varying sizes. One circle is on the left, one is on the right, and a smaller one is in the lower-left area.

SURESOFT

STATIC Enterprise

STATIC Enterprise

STATIC Enterprise(스테틱 엔터프라이즈)는 코딩 규칙 기반의 정적 분석을 수행하는 소스 코드 정적 분석 도구로, 규칙 검사와 실행시간 오류 검사를 중심으로 다양한 도메인의 규칙과 품질 메트릭을 웹 서버 기반의 중앙 관리 환경을 통해 제공합니다.

핵심경쟁력
(차별점/강점)

다양한 도메인 지원



실행시간 오류 검출



SW 품질 메트릭



쉽고 빠른 결함 수정



AS-IS

- 1 실행해야 오류 확인 가능
- 2 도메인에 따라 요구되는 코딩 규칙
- 3 프로젝트의 결함 추이 추적이 어려움
- 4 결함 관리의 어려움
- 5 결함 수정의 어려움



To-BE

- 1 정적 분석으로 실행시간 오류 분석 제공
- 2 다양한 규칙 모음 제공
- 3 프로젝트 관리자 오버 뷰 제공
- 4 다양한 필터링, 상태 관리 제공
- 5 사용자 추천, AI 기반의 수정 가이드 제공

차별점



Performance

분산 컴퓨팅을
활용한 빠른 분석



Management

프로젝트의 전체 추이
및 진척도 제공



Easy

사용자 추천 기반의
수정 가이드 제공



Customizing

고객사 맞춤 규칙 개발
(별도 협의)

강점



국제인증

- Cert. from SGS TÜV
- CWE Compatibility
- Tool Qualification 제공



사용성

- Web기반 UI
- 손쉬운 분석 요청 유틸리티 제공
- Continuous integration 툴 연동



기대 효과

STATIC

- 01 정적 분석 시간 단축
- 02 결함 수정의 편의성
- 03 프로젝트 관리의 용이성



대규모 프로젝트의 정적분석 자동화 도구

Problem

1

Mission Critical
도메인의 코딩 규칙 검사
및 실행시간 오류
(Runtime Error)에 대한
중요도 증가

2

프로젝트
대규모화로 인한
정적 분석 시간 증가

3

프로젝트
개발 인력의 증가로
인한 커뮤니케이션
비용 증가

4

결함 수정의
어려움

Benefit



1

다양한
규칙 세트 제공

2

정적 분석
시간 감소

3

업무 집중도
향상

4

결함 수정의
용이함

실행시간 오류 검출

검출 가능한 결함 (실행시간 오류) 유형



지원 도메인

도메인	지침	지원 내역
자동차	HKMC A/B/SE급 검증 지침	60개 전부 지원
국방	신뢰성 평가 수행 지침 (CWE 658/659/660)	C/C++ 92개 전부 지원 Java 65개 지원 (추가 지원 예정)
그 외 다수 도메인 규칙 지원		

코딩 규칙검사

- 도메인별로 준수해야 하는 코딩 규칙을 자동으로 검사



자동차 MISRA (C, C++) | ISO 26262/ES 95489-23 | AUTOSAR C++14

: 자동차 내장형 시스템 국제표준 코딩규칙
: 자동차 사이버 보안 규칙



보안 CWE 658(C) / 659(C++) / 660(Java)
CERT Secure Coding Standards
행정안전부 시큐어 코딩 가이드 라인



국방 DAPA

: 방위사업청 신뢰성 지침안 코딩 가이드라인 및 국방보안 약점 점검



항공 DO-178C | JSF

: FAA 항공 내장형 시스템 국제표준 코딩 규칙



철도 IEC 62279/EN 50128

: 열차 제어 시스템 SW 국제표준 가이드라인



전기/전자 IEC 61508

: 전기/전자 시스템 SW 국제 표준 가이드 라인



의료 IEC 62304

: Medical device software – Software life cycle processes



원자력 IEC 60880

: 국제 원자력 안정성 표준



조선/해양 IEC 61508

: 전기/전자 시스템 SW 국제표준 가이드 라인



기타 Naming, Coding style 가이드라인

사이버 보안 코딩 규칙

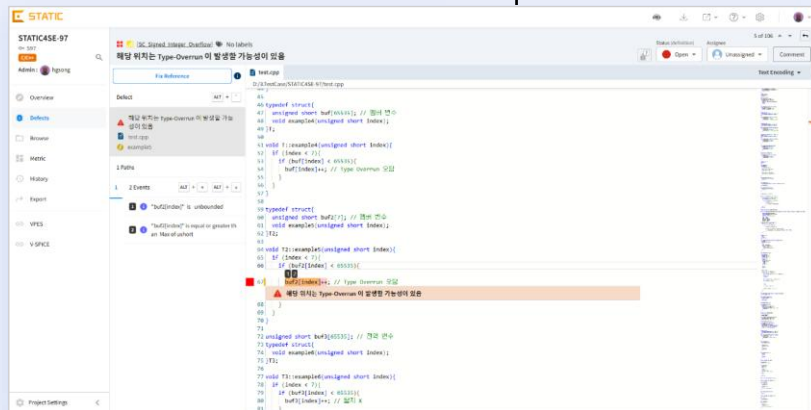
- ES95489-23 100% 대응
- 4종 룰셋 제공을 통해 상세한 결함 정보 파악 가능

보안 코딩 가이드라인 규칙 제공



4종 룰셋 제공

상세한 결함정보
제공



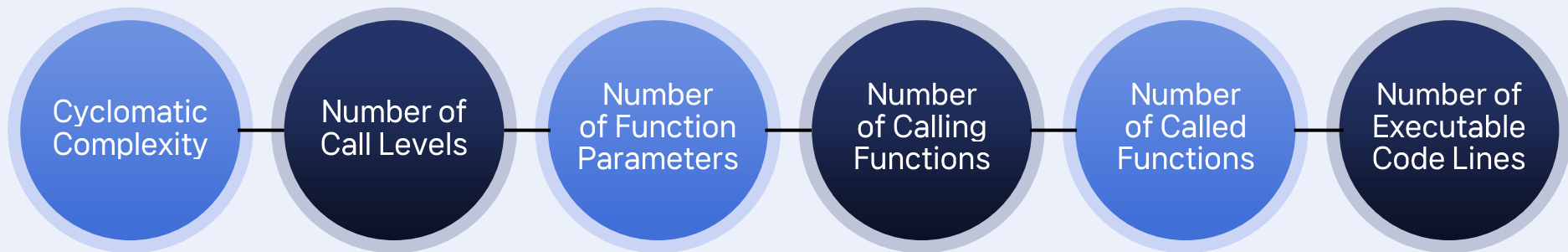
언어별 ES95489-23 지원

언어	전체 룰셋 지원율	심각도 [상] 항목 지원율
C	100%	100%
C++	100%	100%
JAVA	100%	100%

SW 품질 메트릭

- 다양한 SW 품질 메트릭 측정 지원

함수 메트릭 6종 지원 (무기체계 SW 개발 및 관리 매뉴얼 지정 메트릭)



C/C++ 약 25종 메트릭 추가 지원



Java/C#/python/Kotlin 약 15종 메트릭 추가 지원



분산 컴퓨팅을 활용한 빠른 분석

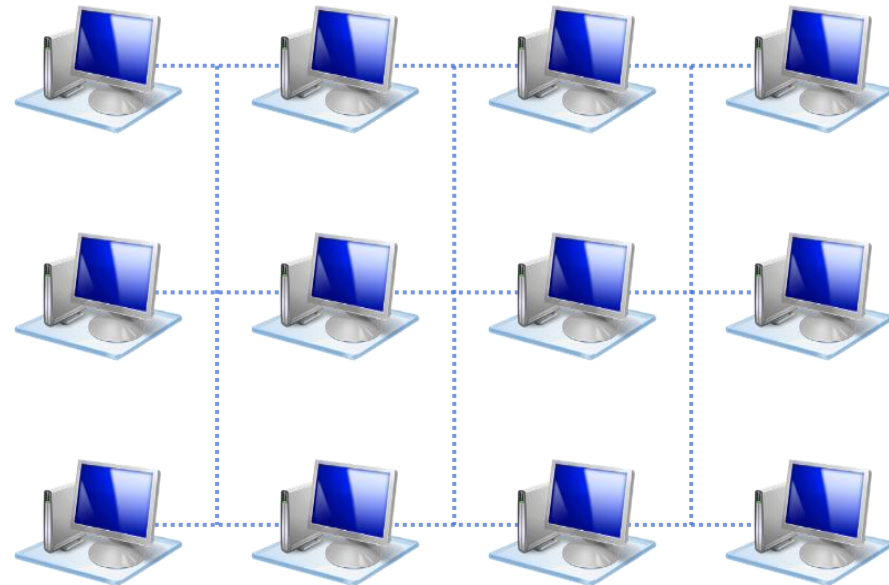
- 여러 PC의 여유 자원을 활용하여 분석에 드는 시간과 비용을 분산
- 약 300개 C++ 소스파일 1분 이내 분석 가능 (별도 APU 서버 확보 필요)

STATIC



분석 요청

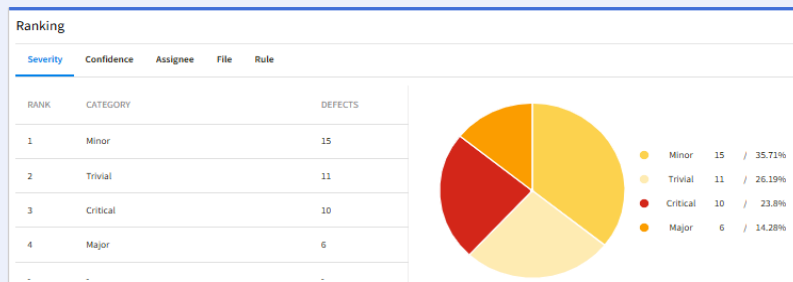
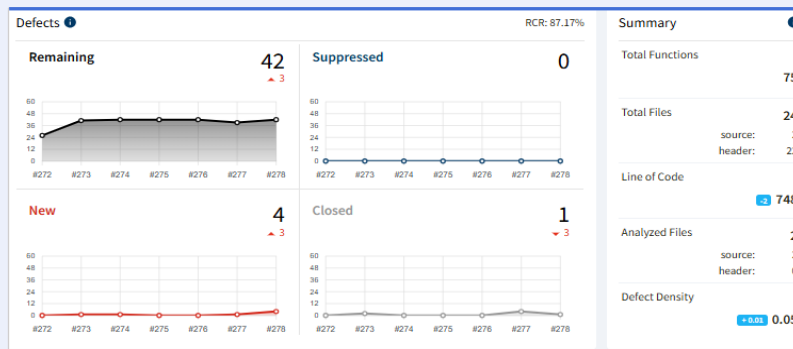
APU Grid



프로젝트 관리자와 실무자의 업무 집중도 향상

- 프로젝트 관리자와 실무자의 관심 영역을 분리한 오버뷰

관리자 영역 (프로젝트 오버뷰)

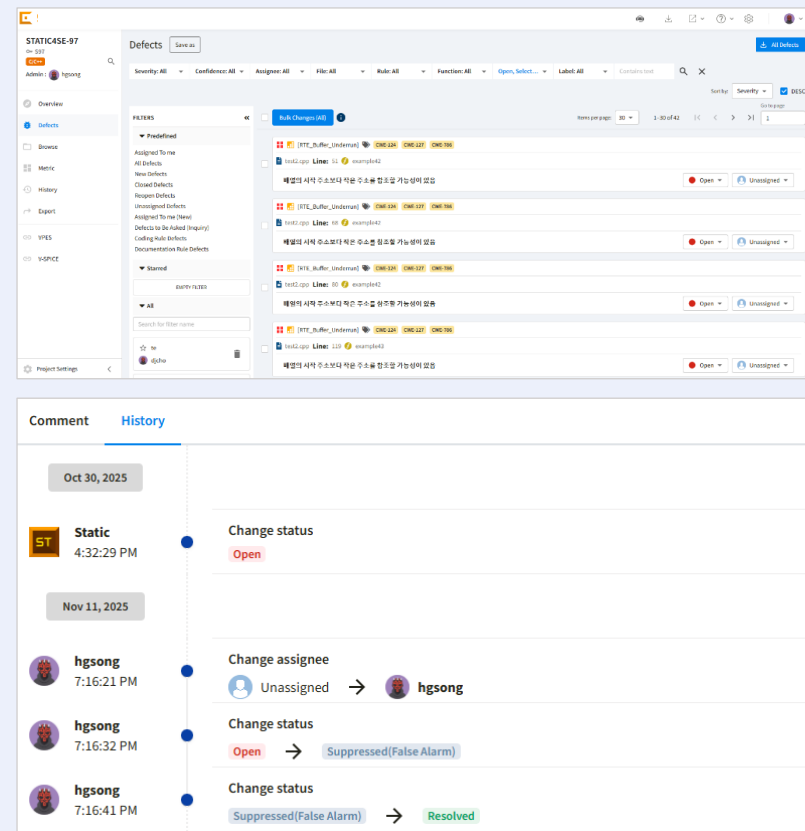
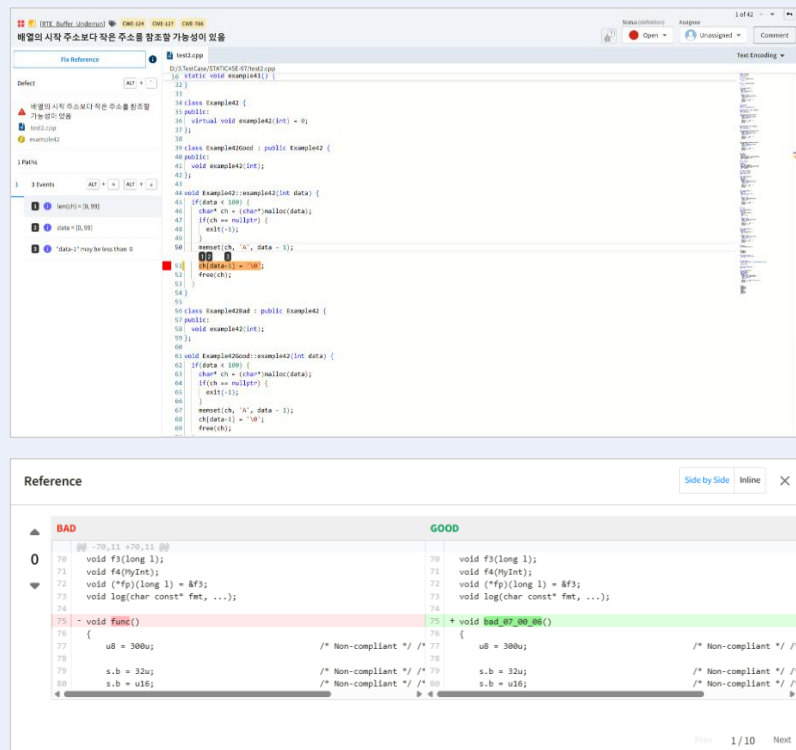


실무자 영역 (결함)



쉽고 빠른 결함 수정 및 관리

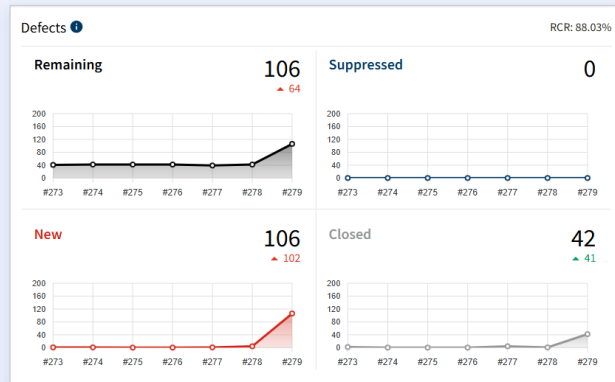
- 결함 위치 확인 및 담당자/상태 변경
- 결함 수정을 위한 가이드 제공
- 결함 필터를 사용한 빠른 검색
- 결함 진행 상황 추적을 통한 관리



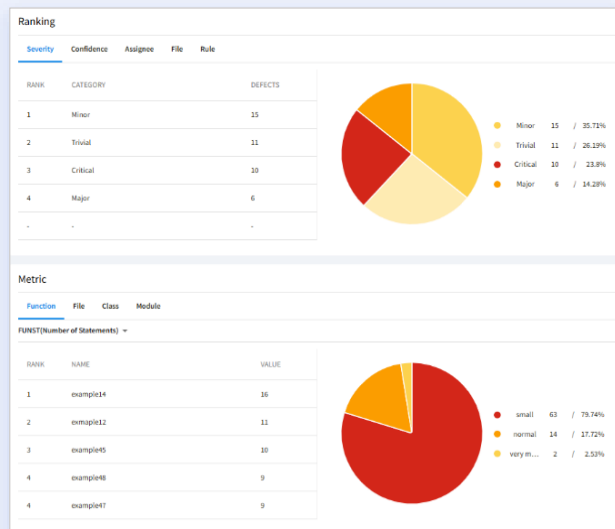
프로젝트 관리자를 위한 오버뷰

- 프로젝트 단위의 정보 제공

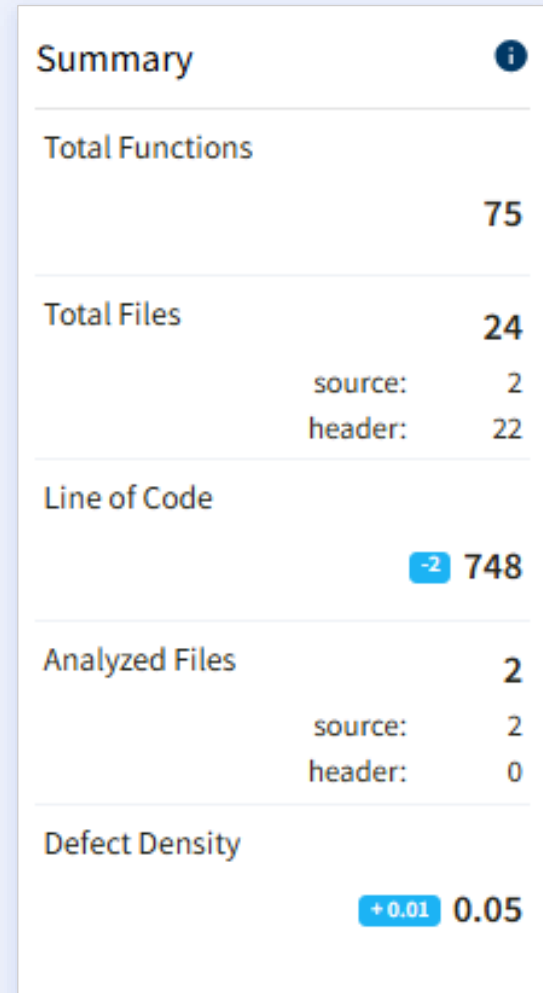
프로젝트 결함 추이 정보



프로젝트 결함 랭킹 정보



분석 결과 요약 정보



프로젝트 관리자를 위한 목표 설정 기능

- 관심 있는 지표별로 프로젝트의 목표 설정
- Burn down 차트를 사용하여 달성도 확인

New Objective

Due date
2025.11.12.

☒ Defects
Please enter a value between 0 and 100,000

☐ Defects by Severity

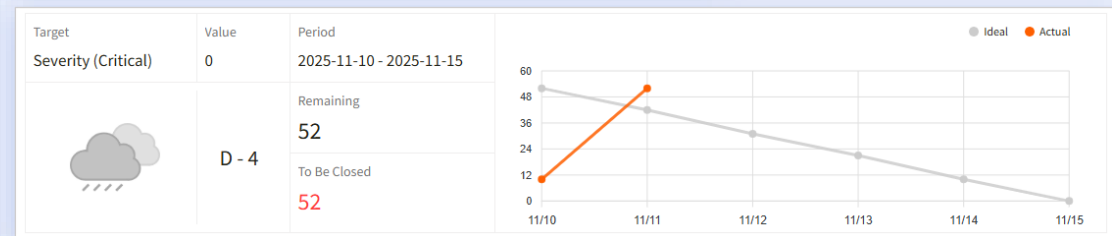
☒ Critical
 0
☐ Major
 0

☐ Minor
 0
☐ Trivial
 0

☐ Others
 0

Submit
Cancel

Due date	2025-11-15	PROGRESS
Target:	Severity (Critical)	Defect: 0
Period:	2025-11-10 ~ 2025-11-15	
Due date	2025-11-12	PROGRESS
Target:	Total	Defect: 0
Period:	2025-11-11 ~ 2025-11-12	
Due date	2025-11-29	PROGRESS
Target:	Severity (Critical)	Defect: 0
Period:	2025-11-11 ~ 2025-11-29	
Due date	2025-12-27	PROGRESS
Target:	Severity (Critical)	Defect: 0
Period:	2025-11-11 ~ 2025-12-27	



결함 Life Cycle & History

- 결함의 Life Cycle을 사용하여 상태 관리
- 결함의 History를 사용하여 진행 상황 관리

Comment History

Oct 30, 2025

static
4:32:29 PM

Nov 11, 2025

hgsong
7:16:21 PM

hgsong
7:16:32 PM

hgsong
7:16:41 PM

Change status
Open

Change assignee
Unassigned → hgsong

Change status
Open → Suppressed(False Alarm)

Change status
Suppressed(False Alarm) → Resolved

New [SC_CPP_Avoid_Read_Uninitialized_Memory]

test.cpp Line: 8

생성자가 없거나, 생성자에서 모든 필드를 초기화하지 않음(no constructor)

Open Unassigned

New [SC_Avoid_Out_Of_Bound_Ptr_Or_Arr_inx]

test.cpp Line: 20 example

배열의 최대 범위보다 큰 주소를 참조할 가능성이 있음

Selected Unassigned

New [SC_Operator_Precedence_Logic_Error] CWE-783

test.cpp Line: 41 example3

연산자 혼용 시 괄호를 사용하지 않았음

Resolved Unassigned

New [SC_Signed_Integer_Overflow]

test.cpp Line: 54 example4

해당 위치는 Type-Overflow 이 발생할 가능성이 있음

Suppressed(Intended) Unassigned

New [SC_Signed_Integer_Overflow]

test.cpp Line: 67 example5

해당 위치는 Type-Overflow 이 발생할 가능성이 있음

Suppressed(False Alarm) Unassigned

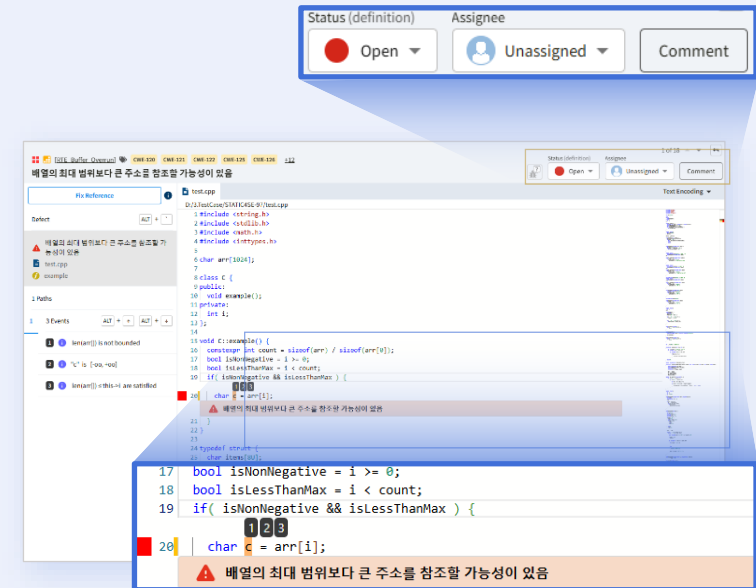
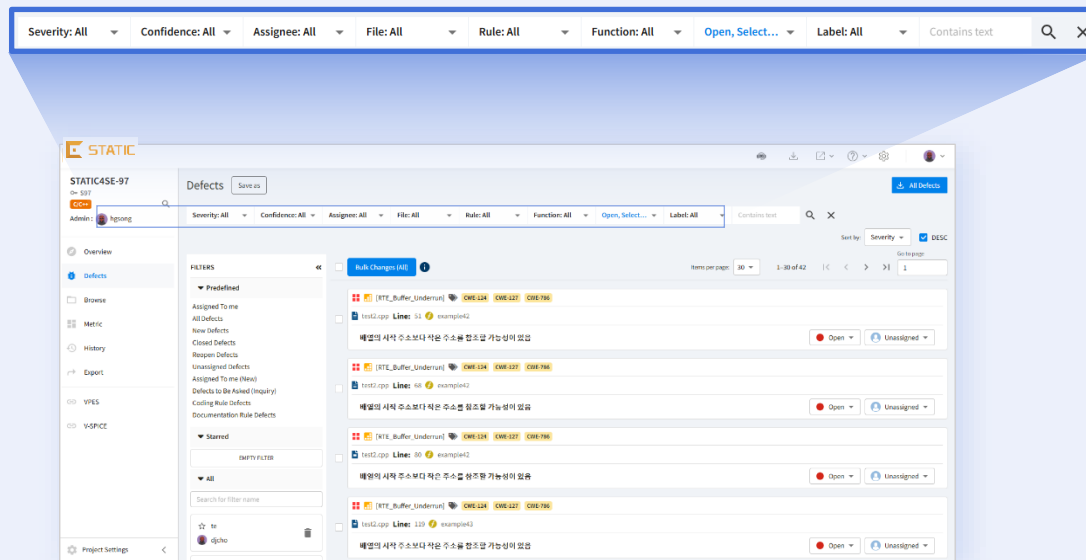
New [SC_Signed_Integer_Overflow]

test.cpp Line: 125 increase

해당 위치는 Type-Overflow 이 발생할 가능성이 있음

Suppressed(Duplicate) Unassigned

다양한 결함 필터 및 결함 확인



- 결함 필터를 사용하여 관심 있는 결함 쉽게 검색
- 결함 상태, 담당자, 규칙, 함수 등 다양한 조합으로 검색
- 결함 위치 확인 및 담당자/상태 변경

- 소스 코드상의 위배 확인
- 결함의 원인 추적을 위한 추가 정보 제공
- 결함 신뢰도 제공

Fix Reference & Fix Assistant

- 사용자 추천 시스템에 의한 수정 이력 추천
- 사용자가 쉽게 결함을 수정할 수 있도록 가이드
- AI를 통한 결함 수정 가이드 (Fix Assistant)



SC_CPP_Check_One_Definition_Rule DCL60-CPP

함수(staticReturnsTrue)를 중복하여 정의함(이전 정의 위치[line:625, file:D:/3.TESTCASE/STATIC4SE-97/TEST.CPP])

Fix Reference

Defect

함수(staticReturnsTrue)를 중복하여 정의함
(이전 정의 위치[line:625, file:D:/3.TESTCASE/STATIC4SE-97/TEST.CPP])

test2.cpp

```
1 #include <string.h>
2 #include <stdlib.h>
3
4 int globalFive = 5;
5
6 static int staticReturnsTrue()
7 {
8     return 1;
9 }
10
11 static int staticReturnsTrue()
12 {
13     return 0;
14 }
```

0 Paths

사용자 추천 버튼

Reference

Side by Side Inline X

BAD GOOD

1

BAD	GOOD
70 void f3(long l);	70 void f3(long l);
71 void f4(MyInt);	71 void f4(MyInt);
72 void (*fp)(long l) = &f3;	72 void (*fp)(long l) = &f3;
73 void log(char const* fmt, ...);	73 void log(char const* fmt, ...);
74	74
75 - void func()	75 + void bad_07_00_06()
76 {	76 {
77 u8 = 300u; /* Non-compliant */	77 u8 = 300u; /* Non-compliant */
78	78
79 s.b = 32u; /* Non-compliant */	79 s.b = 32u; /* Non-compliant */
80 s.b = u16; /* Non-compliant */	80 s.b = u16; /* Non-compliant */

Prev 1 / 10 Next

Troubleshooting guide기반 chatbot

- AI를 통한 chatbot 가이드
- Troubleshooting guide를 학습한 AI



The screenshot displays the STATIC web application interface. At the top, there's a header with the STATIC logo and navigation icons. Below the header, a 'Projects' section lists several projects with their status and analysis results. A chatbot window is open, showing a conversation about the STATIC tool. The chatbot's response explains that STATIC is a static analysis tool that detects errors and vulnerabilities in code, and it lists its features and supported languages.

Project	Remaining	Suppressed	New	Closed	Created time	Last analysis
test	73	3	73	0	Nov 4, 2024, 8:17 AM	Nov 4, 2024, 8:58 AM
ctest	0	12	0	0	Nov 4, 2024, 9:30 AM	Nov 4, 2024, 9:34 AM
testtest	2	120	2	0	Nov 4, 2024, 9:39 AM	Nov 4, 2024, 9:40 AM
test2	0	25	0	0	Nov 4, 2024, 11:52 AM	Nov 4, 2024, 11:56 AM
STATIC4	-	-	-	-	Nov 4, 2024, 3:55 PM	-

Static 도구에 대해서 설명해줘

STATIC은 코드에서 치명적 오류와 잠재적 보안 취약점을 감지하는 정적 코드 분석 솔루션입니다. 코드 가이드라인에 따라 시스템 결함, 신뢰성 저하, 불안정한 조건을 초래하는 심각한 취약점이나 버그를 빠르게 분석하고 검증합니다. 이 제품은 런타임 시 발생할 수 있는 데이터 및 메모리 오류를 감지할 수도 있습니다.

STATIC의 주요 특징으로는 MISRA_C_2004, MISRA_C_2012, CERT_C, CWE_C 등의 코딩 규칙을 준수할 수 있는 기능이 있습니다. 또한, 개발 초기에 치명적인 결함을 감지하여 소프트웨어 동작의 정확한 모델을 만드는 데 도움을 줍니다. STATIC은 프로젝트의 전반적인 품질과 보안을 개선하며, 개발 도중 발견된 결함에 대한 즉각적인 피드백을 제공하므로 개발자의 역량 향상에 기여합니다.

STATIC은 다양한 유형의 Java, C, C++, C# 프로젝트에 대한 수동 또는 자동 정적 분석도 지원하며, 프로젝트 생성을 통해 분

Send message

Legacy Code 분석 제외 기능

- Baseline을 설정하여 특정 시점 이후 발생하는 결함만 관리

Project Settings

Analysis

Rule

Metric

Analysis Scope

Baseline >

Project

General Setting

Objectives

Automatic Assignment

Members

Actions

Vpes Auto Upload

Overwrite State

Baseline

You can set the baseline and check the history.

Set Baseline

Baselines themselves are valued not only to use to identify the notable state of work product(s) but also provide historical views of how work product elements have proceeded together over time. When a historical baseline is retrieved, the state of the work product(s) in that subset share the same significance in their history of changes that allows project leaders to compare the relative progress of single parts of a project to the project as whole, identification, monitoring, and retrieval are critical to the success of configuration management. Once retrieved, the baseline may be compared to a particular configuration or another baseline.

[Set](#)

Activity

Analysis	Date	Defects	Description	Status
#281	Nov 11, 2025, 7:52 PM	18	Unset	Unset
#281	Nov 11, 2025, 7:52 PM	18	Set Baseline	Set

품질 메트릭 제공

- 다양한 단위(모듈, 파일, 클래스, 함수)의 SW 품질 메트릭 (약 30종) 제공
- 메트릭 위배 설정 및 경고 표시

주요 품질 메트릭

- Cyclomatic Complexity
- Maximum Nesting Level
- Maximum Function Calling Number
- Maximum Function Called Number
- Recursion Function Number

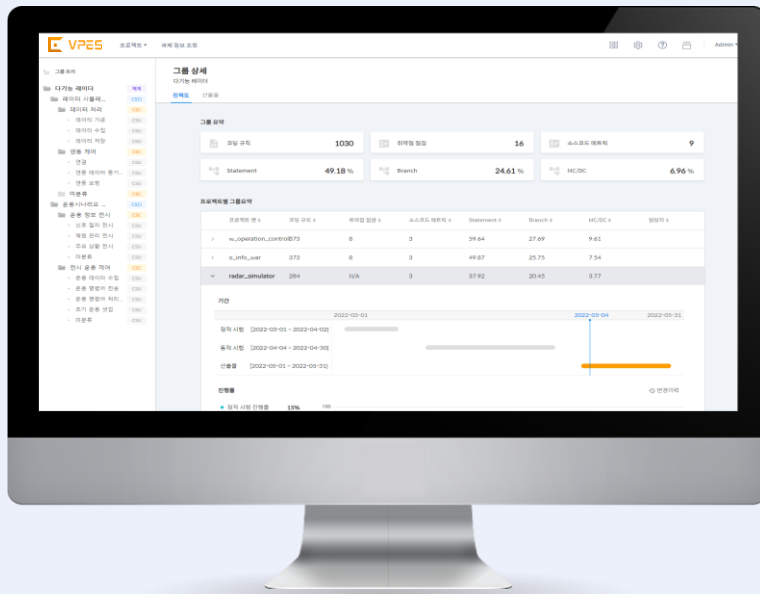
The screenshot displays the SURESOFT quality metrics interface. At the top, there are tabs for 'Module', 'File', 'Class', and 'Function'. Below these, a search bar and filter options (All, Warning, Suppression, Diagnosis Stage) are visible. A table lists various functions with their corresponding metrics. A red box highlights a column of values, and a red arrow points from this column to the 'Diagnosis' section of the configuration panel.

Function Name	FUNST	FUCYC	FUMCYC	FUMNC	FUMIV	FUNSP	FUNDCR	FUNCE	FUNPA	FUNGS	FUNRP
Warning / Suppression	0/0	0/0	0/0	0/0	0/0	0/0	0/0				
bad	3	1	1	-	1:0	1	-		2	-	-
bad3	1	2	2	1	2:0	2	-	1	-	-	
C::example	1	2	2	1	2:1	2	-	-	-	-	
call35	3	1	1	-	1:0	1	-	3	1	-	
call37	1	1	1	-	1:0	1	-	1	-	-	
callToMem	2	1	1	-	1:0	1	-	2	-	-	
D::example25	1	2	2	1	2:1	2	-	-	-	-	
decrease	2	2	2	1	2:0	2	1	-	1	-	
decrease	2	2	2	1	2:0	2	1	-	1	-	
E::example26	1	2	2	1	2:1	2	-	-	-	-	
example10	1	1	1	-	1:0	1	-	1	-	-	
example11	5	4	4	3	4:0	4	-	-	2	-	
example13	6	4	4	3	4:0	4	-	-	2	-	
example14	16	9	9	4	9:0	9	-	-	1	-	
example15	1	1	1	-	1:0	1	-	-	2	-	

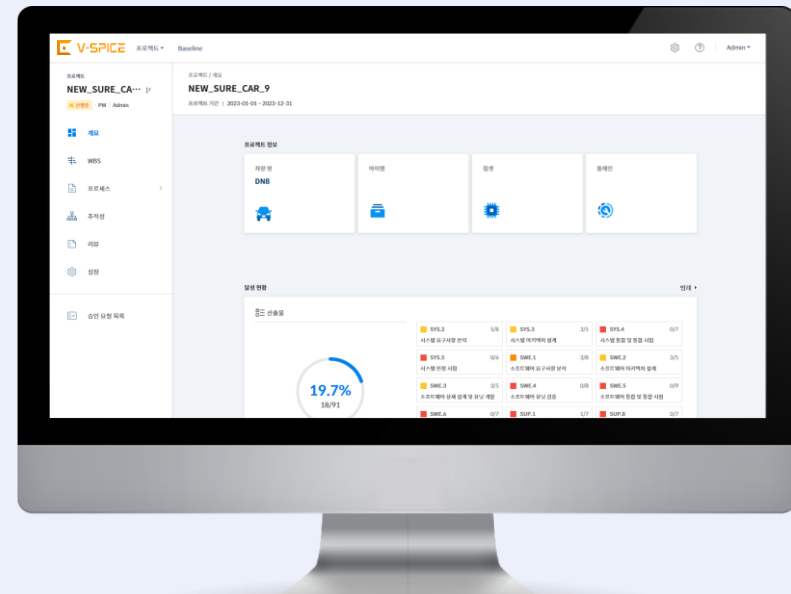
The configuration panel for the 'FUNDNC' metric is shown below. It includes a 'Diagnosis' section with a 'Stage' dropdown set to '2'. A red box highlights the 'Stage' dropdown, and a red arrow points from the 'FUNDNC' column in the table to this dropdown. The 'Diagnosis' section also includes a 'Warning' checkbox and a 'Save' button.

다양한 시험도구와 연동가능

- 정적시험 자동화 환경 구축 시 자사도구 VPES, V-SPICE와 연동가능
- 지속 통합 시스템(Jenkins 등)과 연동



VPES (빌드 및 시험 자동화 도구)



V-SPICE (프로세스 리포팅 자동화 도구)

지원 환경

구분		세부 사항
언어지원		C/C++, Java, C#, Kotlin, Python
지원환경	OS	▪ Client : Windows 10 이상, Ubuntu OS (기타 Linux 환경은 문의) ▪ Server(Was)/Agent(APU) : Windows 10 이상
	컴파일러 지원	▪ ARM 계열 ▪ Freescale 계열 ▪ GreenHills 계열 ▪ Keil 계열 ▪ Renesas 계열 ▪ 기타 100개 이상 지원 ▪ 신규 툴체인 2주 이내 지원 가능

하드웨어 권장 사양

구분	CPU	RAM	HDD
Server (Was)	8 Core 3.0GHz 이상 64bit	16GB 이상	약 1T의 여유 공간
Agent (APU)	8 Core 3.0GHz 이상 64bit	32GB 이상	약 1T의 여유 공간



ISO 26262 검증 적용사례

| 차량 부품 제조사(H사)

상황

실제 오류를 차후 방지하기 위해 맞춤 코딩 규칙 필요

해결 방안
및 결과

- 오류를 분석하여 맞춤 코딩 규칙 개발 및 적용
- 개발프로세스에 개발된 코딩 규칙 검증 포함

※ 적용 분야 : 엔진 설계 (C언어)



IEC 61508 검증 적용사례

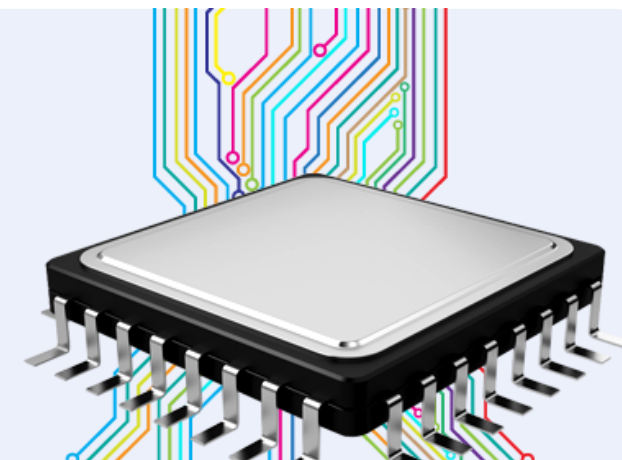
| 반도체 제조사(S사)

상황

별도의 정적 분석 도구를 사용하지 않음

해결 방안
및 결과

- 빌드 자동화 서버와 연동하여 효율적으로 SW 품질관리가 가능해짐
- MISRA 코딩 규칙을 적용하여 SW의 품질 향상



무기체계 내장형 SW 신뢰성 검증 적용사례

| 방산시스템 개발사(사)

검증대상

차기 전술 유도무기 발사통제 장비 사격통제 콘솔용 GUI SW

고객사 요구사항

SW 신뢰성 시험 요구사항 만족

요구사항	정적 시험	동적 시험
정의	SW 실행 없이 소스 코드 내 결함 검출	요구사항 기반의 테스트 케이스로 실행 확인
시험 항목	코딩 규칙 검출 실행시간 오류 검출	코드 실행률 측정

도구 적용

① 실행시간 오류 검출 | ② 신뢰성 평가 수행지침 기반 규칙으로 결함을 검출하고 수정

결과

SW 신뢰성 시험기준 만족 | LIG 넥스원 검수 기준 만족

The background is a solid dark blue. It features decorative elements consisting of several thin, light blue wavy lines that sweep across the lower half of the slide. Three solid blue circles of varying sizes are also present: a medium-sized one on the left, a small one at the bottom left, and a larger one on the right.

SURESOFT

STATIC Standalone

STATIC Standalone

STATIC Standalone(스테틱 스탠드얼론)은 코딩 규칙 기반의 정적 분석을 수행하는 소스 코드 정적 분석 도구로, 규칙 검사와 실행시간 오류 검사를 중심으로 다양한 도메인의 규칙과 품질 메트릭을 VS Code 기반의 Standalone IDE를 통해 제공합니다.

핵심경쟁력
(차별점/강점)

AI Chat



실행시간 오류 검출



SW 품질 메트릭



쉽고 빠른 결함 수정



AS-IS

- 1 실행해야 오류 확인 가능
- 2 제한된 위배 수정 안내
- 3 위배 확인과 수정의 APP이 다름
- 4 결함 관리의 어려움
- 5 정적 분석 설정의 어려움



To-BE

- 1 정적 분석으로 실행시간 오류 분석 제공
- 2 AI를 통한 위배 수정 가이드
- 3 하나의 APP에서 위배 확인과 수정을 모두 수행
- 4 다양한 필터링, 상태 관리 제공
- 5 개발 환경 별 자동화된 분석 정보 추출

차별점



Convenience

미리 준비된 분석 설정을 통한
틀체인 정보 자동 추출



UX

VS Code Guidelines
기반으로 구성된 UI/UX



Easy

사용자 추천 기반의
수정 가이드 제공



Customizing

고객사 맞춤 규칙 개발
(별도 협의)

강점



- Cert. from SGS TÜV
- CWE Compatibility
- Tool Qualification 제공



- VS Code기반 UI
- 손쉬운 분석 요청 유틸리티 제공
- AI를 통한 질의 응답



기대 효과

STATIC

01 소스코드 결함 조기 발견

02 정적 분석 시간 단축

03 결함 수정 생산성 향상



개발자를 위한 정적분석 자동화 도구

Problem

1

Mission Critical
도메인의 코딩 규칙 검사
및 실행시간 오류
(Runtime Error)에 대한
중요도 증가

2

개발 환경별
정적분석 설정에
기술지원 필요

3

출장지에서
정적분석 수행이
어려움

4

결함 수정의
어려움
(multi context)

Benefit



1

다양한
규칙 세트 제공

2

미리 준비된 틀체인 설정과
분석 설정 시간 감소

3

저사양 환경 지원과
동글 라이선스 지원

4

결함 수정의
용이함
(single context)

실행시간 오류 검출

검출 가능한 결함 (실행시간 오류) 유형



지원 도메인

도메인	지침	지원 내역
자동차	HKMC A/B급 검증 지침	48개 전부 지원
국방	신뢰성 평가 수행 지침 (CWE 658/659)	C/C++ 92개 전부 지원
그 외 다수 도메인 규칙 지원		

코딩 규칙검사

- 도메인별로 준수해야 하는 코딩 규칙을 자동으로 검사



자동차 MISRA (C, C++) | ISO 26262/ES 95489-23 | AUTOSAR C++14

: 자동차 내장형 시스템 국제표준 코딩규칙
: 자동차 사이버 보안 규칙



보안 CWE 658(C) / 659(C++)
CERT Secure Coding Standards
행정안전부 시큐어 코딩 가이드라인



국방 DAPA

: 방위사업청 신뢰성 지침안 코딩 가이드라인 및 국방보안 약점 점검



항공 DO-178C | JSF

: FAA 항공 내장형 시스템 국제표준 코딩 규칙



철도 IEC 62279/EN 50128

: 열차 제어 시스템 SW 국제표준 가이드라인



전기/전자 IEC 61508

: 전기/전자 시스템 SW 국제 표준 가이드 라인



의료 IEC 62304

: Medical device software – Software life cycle processes



원자력 IEC 60880

: 국제 원자력 안정성 표준



조선/해양 IEC 61508

: 전기/전자 시스템 SW 국제표준 가이드 라인



기타 Naming, Coding style 가이드라인

사이버 보안 코딩 규칙

- ES95489-23 100% 대응
- 4종 룰셋 제공을 통해 상세한 결함 정보 파악 가능

보안 코딩 가이드라인 규칙 제공

Suresoft_CPP_Secure_Coding
슈어소프트 C++ 언어용 코딩 가이드...

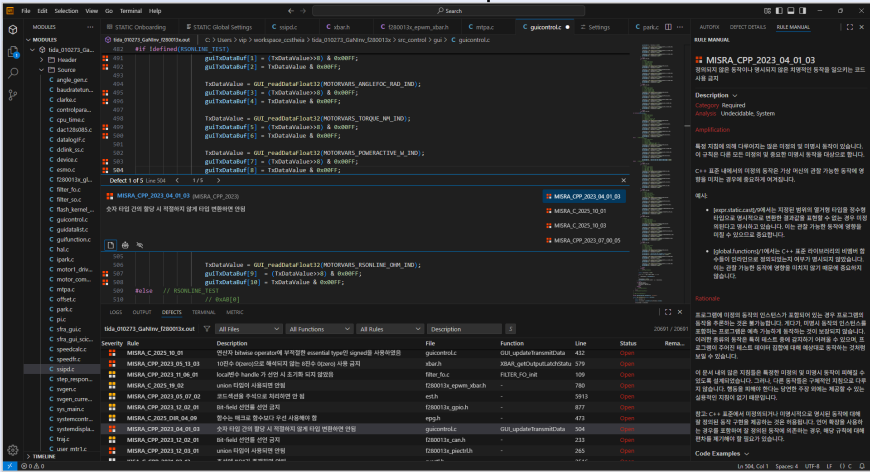
Suresoft_C_Secure_Coding
슈어소프트 C 언어용 코딩 가이드라...

Suresoft_CPP_Secure_Codin...
슈어소프트 C++ 언어용 코딩 가이드...

Suresoft_C_Secure_Coding_F...
슈어소프트 C 언어용 코딩 가이드라...

4종 룰셋 제공

상세한 결함정보 제공



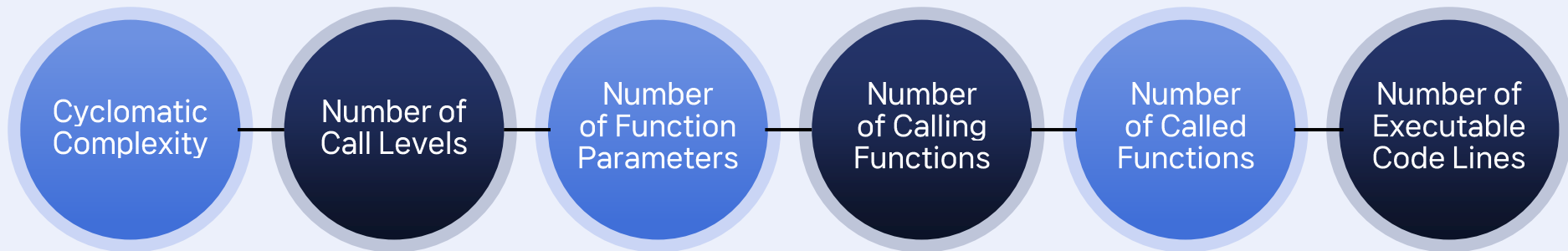
언어별 ES95489-23 지원

언어	전체 룰셋 지원율	심각도 [상] 항목 지원율
C	100%	100%
C++	100%	100%

SW 품질 메트릭

- 다양한 SW 품질 메트릭 측정 지원

함수 메트릭 6종 지원 (무기체계 SW 개발 및 관리 매뉴얼 지정 메트릭)



C/C++ 20종 메트릭 추가 지원

모듈	파일	함수
① Number of files	Physical Line of code Line of code Comment ratio	• Number of statements
② Number of files(source)		• Modified cyclomatic complexity
③ Number of files(header)		• Maximum nesting depth
④ Number of function		• Myer's interval
⑤ Number of Recursions		• ...

Standalone 형 정적 분석 제품

- 소스코드가 편집 가능한 제품으로 위배 결과를 확인과 수정을 하나의 Context에서 수행
- VS Code Guidelines 기반으로 구성된 UI/UX 로 낮은 학습 비용

소스코드 수준 상세 결함 정보 표시

The screenshot displays the Suresoft static analysis tool interface, which is designed to look like a standard code editor (VS Code). It features a sidebar on the left for navigating through project modules and source files. The main area shows the source code of a file named 'gui.c'. A defect list is visible at the bottom, listing various issues found, such as MISRA_C_2025_10_01 and MISRA_CPP_2023_04_01_03. A detailed view of a specific defect is shown on the right, including its description, category, and rationale.

Module / Header / Source

Defect / Metric Detailed Information Display

Rule Manual

- 룰 설명
- 위배상세
- AI Chat
- AutoFix

모듈 / 헤더 / 소스

위배/메트릭 상세 정보 표시

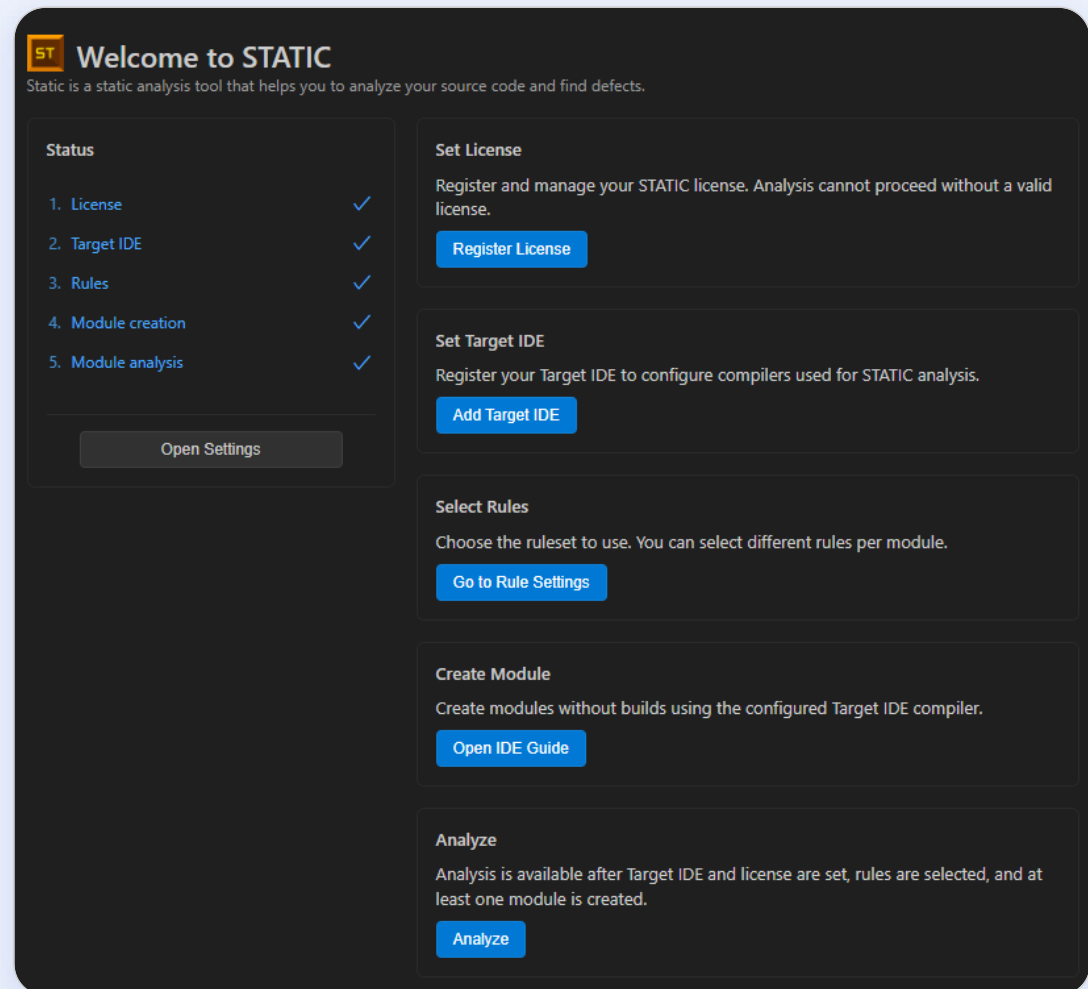
간편한 설치와 Onboarding

- 설치부터 정적 분석 결과 확인까지 쉽고 빠르게 수행
- Onboarding(Welcome to STATIC)을 통해 처음 사용하는 사용자도 쉽게 사용

Onboarding

- ① [STATIC Standalone(이하 [ST])] 설치
- ② [ST] 실행
- ③ [ST] 라이선스 인증
- ④ [ST] TARGET IDE 설정 (컴파일러 경로 입력)
- 대상 IDE quick guide 열기
- ⑤ [ST] 사용할 룰셋/룰 선택
- ⑥ [대상 IDE] wrapper 설정
- ⑦ [대상 IDE] 빌드
- ⑧ [ST] 분석
- ⑨ 결과 확인

< STATIC Enterprise 대비 70% 절차 간소화 >



Target IDE 별 Quick Guide

- 개발 환경별 Quick Guide를 배포하여 비개발자도 쉽게 사용
- 미지원 환경에 대해서도 지원 Tool을 제공하고, 지속적으로 정식 지원 범위 확대

지원 Tool 제공

▼ 한글

-  [Code Composer Studio \(Eclipse\)](#)
-  [Code Composer Studio \(Theia\)](#)
-  [Microchip Studio](#)
-  [MPLAB X IDE](#)
-  [Vitis Unified](#)
-  [STM32CubeIDE](#)

▼ 영어

-  [Code Composer Studio \(Eclipse\)](#)
-  [Code Composer Studio \(Theia\)](#)
-  [Microchip Studio](#)
-  [MPLAB X IDE](#)
-  [Vitis Unified](#)
-  [STM32CubeIDE](#)

환경별 Quick Guide

적용 환경

OS

- Windows 11

IDE

- Code Composer Studio 12.8.1

Compiler 버전

- ti-cgt-c2000_22.6.1.LTS

Compiler 경로

- C:\ti\ccs1281\ccs\tools\compiler\ti-cgt-c2000_22.6.1.LTS\bin

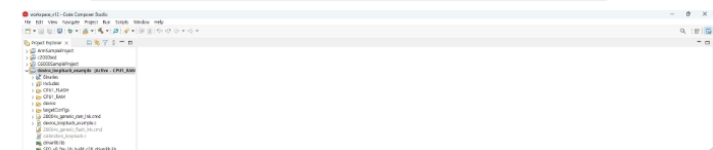
Compiler 이름

- cl2000.exe

툴체인 생성

원본 컴파일러 확인 방법

- 원본 프로젝트 우클릭 → [Rebuild Project] → [Console]에 출력된 컴파일러, 링커 경로 확인
 - 예시: `C:\ti\ccs1281\ccs\tools\compiler\ti-cgt-c2000_22.6.1.LTS\bin\cl2000`



Vibe Inspection

- AI chat을 통해 자연어로 정적분석 프로세스 수행
- 모듈 단위 정밀 분석을 통해 검출 정확도 향상과 단일 위배/파일/모듈 단위 Autofix 기능 제공



주요기능

- 자연어 기반 소통
- 위배 설명
- 위배 수정
- 오탐 판정
- 정적 분석

The screenshot displays the SURESOFT Vibe Inspection tool interface. The main window shows a C source file 'transmitter.c' with a defect highlighted at line 326. A table below lists several defects, including MISRA_C_2025_12_01 and MISRA_CPP_2023_08_00_01. On the right, a 'CHAT' panel shows a conversation about a defect, and a 'Command' panel shows a command input field. A large white arrow labeled '상호 작용' (Interaction) points from the chat/command area towards the defect details.

소스코드 수준 상세 결함 정보 표시

상호 작용

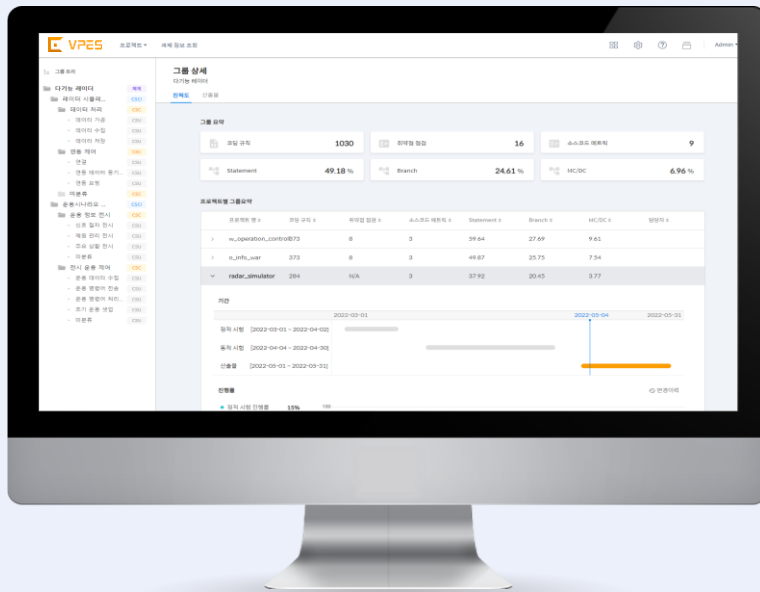
Severity	Rule	Description	File	Function	Line	Status
High	MISRA_CPP_2023_08_00_01	우선 순위가 다른 연산자(+, *)를 괄호 없이 혼용함	device.c	Device_enabl	346	Open
High	MISRA_C_2025_12_01	코드색션을 주석으로 처리하면 안 됨	device.h	-	512	Open
High	MISRA_C_2025_12_01	우선 순위가 다른 연산자(+, *)를 괄호 없이 혼용함	device.h	-	512	Open
High	MISRA_C_2025_12_01	__DEVICE_H__는 #define 가 금지된 매크로임	device.c	-	409	Open
High	MISRA_C_2025_12_01	unsigned type의 상수(0xFFFFFFFF)에는 U suffix를 사용	device.c	-	409	Open
High	MISRA_CPP_2023_08_00_01	숫자 타입 간의 할당 시 적절하지 않게 타입 변환하면 안 됨	transmitter.c	initEPWM3	554	Open
High	MISRA_CPP_2023_08_00_01	숫자 타입 간의 할당 시 적절하지 않게 타입 변환하면 안 됨	device.c	Device_enabl	351	Open
High	MISRA_CPP_2023_08_00_01	숫자 타입 간의 할당 시 적절하지 않게 타입 변환하면 안 됨	device.c	Device_enabl	300	Open
High	MISRA_CPP_2023_08_00_01	숫자 타입 간의 할당 시 적절하지 않게 타입 변환하면 안 됨	device.c	Device_enabl	345	Open
High	MISRA_CPP_2023_08_00_01	숫자 타입 간의 할당 시 적절하지 않게 타입 변환하면 안 됨	device.c	Device_enabl	386	Open

정량 측정

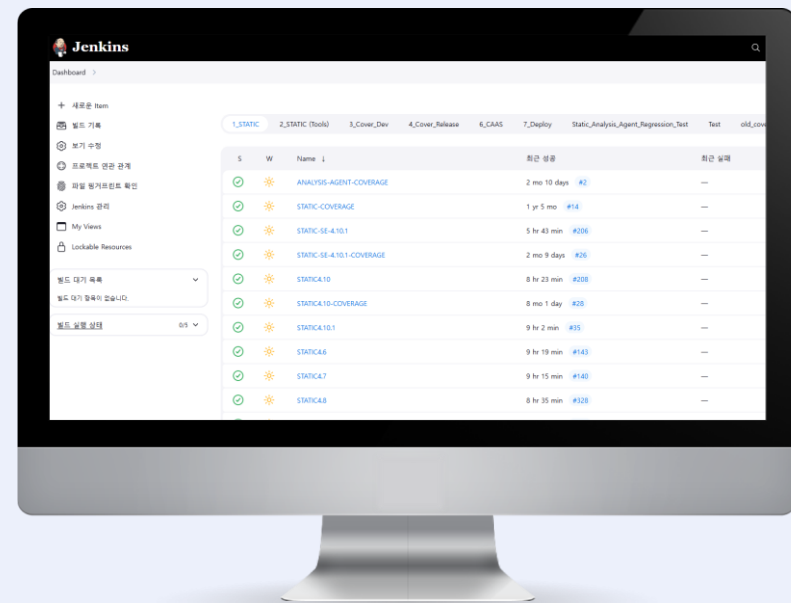
- ALIRA-AI(QWEN3) 기준 - 82.7% 위배 자동 수정 (150개 케이스에서 124개 수정)
- ALIRA-AI(QWEN3) 기준 - 88% 오탐 식별 (50개 케이스에서 44개 판정)

다양한 시험도구와 연동

- 정적시험 자동화 환경 구축 시 자사 도구 VPES와 연동
- 지속 통합 시스템(Jenkins 등)과 연동



VPES (빌드 및 시험 자동화 도구)



Jenkins (지속 통합 시스템)

지원 환경

구분		세부 사항
언어지원		C/C++
지원환경	OS	Windows 10 이상
	지원 IDE와 컴파일러	- Code Composer Studio 5 ~ 12 (cl2000, C6000) - Code Composer Studio – theia 20 (cl2000, C6000) - Microchip Studio 7 (xc8, xc8_avr, xc16, xc32, arm_gcc, avr8, avr32) - MPLAB X IDE 6 (xc8, xc8_avr, xc16, xc32, arm_gcc, avr8, avr32) - STM32CubeIDE 1 (arm-gcc) - Vitis Unified v2025.1.0 (arm_gcc, arm64_gcc, mb_gcc) - Makefile (gcc 3.x ~ 14.2.0) - Manual Mode (툴체인 수동 설정) - 신규 환경 지속적 지원 확대

하드웨어 권장 사양

구분	CPU	RAM	HDD
권장 사양	8 Core 3.0GHz 이상 64bit	8 GB 이상	약 100GB의 여유 공간
최소 사양	4 Core 3.0GHz 이상 64bit	16 GB 이상	약 1TB의 여유 공간



ISO 26262 검증 적용사례

차량 부품 제조사(H사)

상황

해결 방안
및 결과

실제 오류를 차후 방지하기 위해 맞춤 코딩 규칙 필요

- 오류를 분석하여 맞춤 코딩 규칙 개발 및 적용
- 개발프로세스에 개발된 코딩 규칙 검증 포함

※ 적용 분야 : 엔진 설계 (C언어)



IEC 61508 검증 적용사례

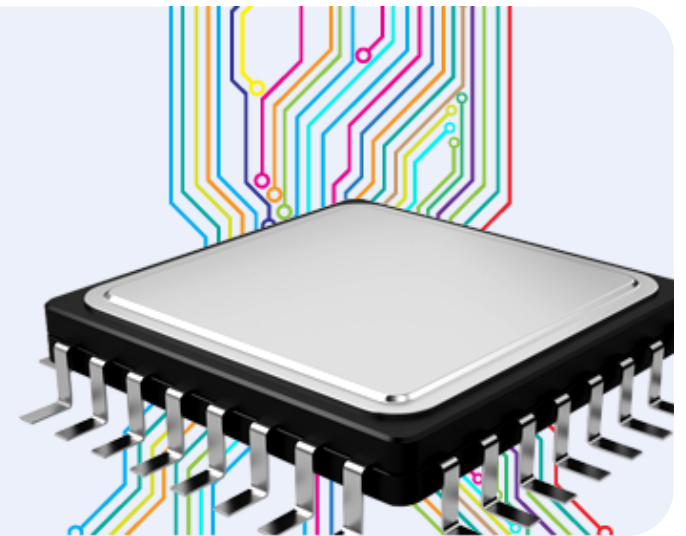
반도체 제조사(S사)

상황

해결 방안
및 결과

별도의 정적 분석 도구를 사용하지 않음

- 빌드 자동화 서버와 연동하여 효율적으로 SW 품질관리가 가능해짐
- MISRA 코딩 규칙을 적용하여 SW의 품질 향상



무기체계 내장형 SW 신뢰성 검증 적용사례

| 방산시스템 개발사(사)

검증대상

차기 전술 유도무기 발사통제 장비 사격통제 콘솔용 GUI SW

고객사 요구사항

SW 신뢰성 시험 요구사항 만족

요구사항	정적 시험	동적 시험
정의	SW 실행 없이 소스 코드 내 결함 검출	요구사항 기반의 테스트 케이스로 실행 확인
시험 항목	코딩 규칙 검출 실행시간 오류 검출	코드 실행률 측정

도구 적용

- 실행시간 오류 검출
- 신뢰성 평가 수행지침 기반 규칙으로 결함을 검출하고 수정

결과

- SW 신뢰성 시험 기준 만족
- LIG 넥스원 검수 기준 만족



Thank You