

Static analysis tool C-STAT

Ensuring code quality through static analysis

Catch issues before they become problems with C-STAT, the static analysis tool fully integrated into the IAR Embedded Workbench IDE. Effortlessly ensure your code complies with MISRA standards and hundreds of security checks from CWE and CERT, all without leaving your development environment.

Key Highlight Features

• Analysis of C and C++ code

• Includes more than 1000 checks in total, some comply with rules as defined by MISRA C:2023, MISRA C:2012, MISRA C++:2008 and MISRA C:2004

- More than 250 checks mapping to issues covered by CWE, SANS Top25 and OWASP
- Checks compliance with the coding standard CERT C for secure coding
- Fully integrated with the IAR Embedded Workbench IDE and the command line IAR Build Tools
- Comprehensive and detailed error information
- Fast execution
- Available for most IAR Embedded Workbench and IAR Build Tools products
- Also available as TÜV SÜD certified version for selected IAR functional safety editions

C-STAT checks code compliance with industry standards MISRA, CWE and CERT C/C++

C-STAT checks compliance with MISRA rules, CERT C/C++ Secure Coding Standards, and CWE-defined weaknesses, simplifying adherence to these standards with actionable insights and industry-aligned outputs.

MISRA (the Motor Industry Software Reliability Association) rules, initially developed for the automotive industry, are now widely used across embedded systems. MISRA-C:2023 builds on MISRA-C:2012 (and amendments) with enhanced rules for modern safety and security challenges. The rules are classified as mandatory, required, or advisory, covering areas like compiler differences, avoiding unsafe functions, limiting code complexity, and ensuring maintainability.

CWE (the Common Weakness Enumeration) is a community-developed dictionary of software weaknesses, detailing potential consequences, mitigations, code samples, and references to help manage vulnerabilities.

CERT provides secure coding rules for C and C++ to prevent coding and design errors that can lead to vulnerabilities. Each guideline includes a title, description, non-compliant code example, and compliant solutions, aimed at eliminating insecure practices and undefined behaviors.

> CodeAnalysis - IAR Embedded Workbe	ench IDE -	- Arm	9.60.3						- 0 ×
File Edit View Project Ljet Tools	Windo	w I	ielp						
nn 🖬 🖉 🗛 🕹 👘 nn 🛯 🗧	D C			- < Q > \$ +E < Q > A	D D D - D D D D D D D D D D D D D D D D				
Workspace	• a ×	m		20.1 system_stm3214ss.c c_state* x		C-STAT Messages			* 0
CodeAnalysis - Debug	~	CER	(3(const char *)		fa	Severity High V Filter V content>			
Files	٥		32 enum { 00FF	<pre>tensize = 24 }; const char "msm) {</pre>		Message		Check	Severity
FI CodeAnalysis				r "error log: //error	log has not been initialized.	I - ■ A Variable `error log' may be uninitialized		CERT-EXP33-C a	High
- CodeAnalysis - Debug	~		35 char buff	er[BUFFERSIZE]; //an in	determinate value is read	Overrun in sprintf writing <unknown size=""> bytes into ta</unknown>	rnet size		
		4		uffer, "Error: %s", error_log);					High
He Dc stat.c		4	37 printf(*%	is\n", buffer);		Variable 'error log' is uninitialized		SPC-uninit-var-all	High
He main.c			39			Format string does not include size of string being con	sumed	CERT-STR31-C a	High
- 🖂 📫 board				c(int i, int 1):		-# A String 'buffer' may not be null-terminated		CERT-STR32-C	High
-Baliar stm32f429ii aca.c			41			- Calling standard library function 'printf' without detect	ing and	CERT-ERR33-C c	High
He stm32f4xx_it.c			42 int glob;			More than one arguments for the function call c(a(),	b())' are	CERT-EXP30-C_b	Medium
- 🕀 🖬 startup			45 = int a(void)	2 Help for IAR Embedded Workbench for	A	- D X		CERT-EXP36-C_b	Medium
HE startup_stm32f429x.s			44 L return g	Thep for watembedded workbench for	r Alm 9.00.X		or pass	CERT-MEM31-C	High
L⊞ System_stm32f4xx.c			45 46 I int b(void)		a		be out	ARR-inv-index-ptr-pos,CER.	Medium
- 🛱 💼 StdPeriph_Driver			47 glob = 42	Hide Locate Back Forward Home	Pivt			CERT-EXP33-C_b	High
HB III Library			45 - return glob				sed as	ATH-div-0-unchk-global,CE.	Medium
Library.a			49	Contents lodex Search Favorites	•			ATH-overflow	Medium
H stm32f4xx_gpio.c		4	50 void cert4	Type in the keyword to find:	(GR)		4 when	CERT-INT31-C_a	Medium
- B readme.txt			51					RED-unused-assign	Low
🛏 🖬 🖬 Output			52 53 uint5 t "hi	SEC-BUFFER-sprintf-overun				CERT-EXP33-C_a	High
- 🕀 🌒 Library - Debug	~		53 uint8_t *h) 54日 (SEC-RUFFER-exercite oversion	C-STAT checks : Descriptions of checks : \$	SEC-BUFFER-sprintf-overrun		SPC-uninit-var-some	High
			55 T return no	SEC-BUFFER-etd-eart-overrun				CERT-EXP33-C_a	High
		-	56 L }	SEC BUFFER and out overun pos SEC BUFFER atmat overun				SPC-uninit-var-some	High
			57	SEC-BUFFER-streat-overtun-pos					
			58 int32_t g(v	SEC-BUFFER-strcpv-syleman	SEC-BUFFER-sprintf-ov	errun		CERT-EXP19-C	Medium
			59日(SEC-BUFFER stropy overrun pos SEC-BUFFER strops overrun				CERT-EXP19-C	Medium
		4	60 uint8_t 1 61 return pl	SEC-RUFFER-stmost-oversio-oos	Synopsis			CERT-EXP33-C_a	High
		4	62 Preturn pl	SEC-BUFFER-stmonp-overtun	A call to the sprintf function will overrun	the target buffer		SPC-uninit-var-some	High
			63 //cstat IA	SEC-BUFFER atmomp overrun pos SEC-BUFFER atmomp-overrun	A car to the spine kineson we overen	ale larger correr.		CERT-EXP33-C_a	High
				SEC-BUFFER-stmcpy-overun-pos	Enabled by default			SPC-uninit-var-some	High
			65 int j=8;	SEC-BUFFER-tainted-alloc-size	· · · · · · · · · · · · · · · · · · ·			CERT-EXP19-C	Medium
				SEC-BUFFER-tainted-copy SEC-BUFFER-tainted-copy	Yes			CERT-EXP19-C	Medium
			67 曱 {	SEC-BUFFER-tainted-rodex				DEFINE-hash-multiple	Medium
			65 int32_t i	SEC-BUFFER-tainted-offset	Severity/Certainty		DE_AE	CERT-MSC41-C_b	High
			70 1-1/1	SEC-BUFFER use-after-free-all SEC-BUFFER use-after-free-some				CERT-MSC41-C_b	High
		-	71 - 1	SEC-BUFFER use-after free-some SEC-DIV-0-compare-after			1_0'	CERT-MSC41-C_b	High
			72	SEC-DIV-0-compare-before	High/High		1	CERT-MSC41-C_b	High
				SBC-DIV-Otainted	rigariga			CERT-MSC41-C_b	High
			74 무 {	SEC-FILEOP open no-close SEC-FILEOP opth-traversal	Full description			CERT-MSC41-C_b	High
			75 int32_t 1 76 uint8 t d	SEC-FILEOP-use-attendiose				CERT-MSC41-C_b	High
			76 uint8_t 0	SEC-INJECTION-exp		the target buffer. Consider using a function that allows you to set the	Config'	CERT-MSC41-C_b	High
		4		SEC-INJECTION/spath SEC-LOOP tainted-bound		ively, you might be able to compare the lenghts of the source and		CERT-MSC41-C_b	High
		1	79 ch = (uir	SEC-NULL-assignment	destination buffer before calling sprintf	This check is identical to LIB-sprintf-overrun.			
			80 L)	SEC-NULL-assignment-fun-pos			NC_AE	CERT-MSC41-C_b	High
			81	SECNULL croated	Coding standards) · ·
Overview CodeAnalysis Library		4		SEC-NULL-onp-bet-fun	CERT STR31-C				
Beady				SEC-NULL-Iteral-pos				In 36 Col 1 System C	AP NUM OVE
				second byte (assembler operator)	III Outstanting that obvious for obviou	on has sufficient enses for character data and the null terminater		Synth C	

iar

Fully integrated into the IDE

C-STAT is fully integrated into the IAR Embedded Workbench IDE, making it as easy to use as standard build tools, with no need for complex setups. It checks your code against MISRA, CERT, and CWE standards, ensuring safety and security by referencing all relevant rules. Since these standards complement each other, C-STAT allows you to check against entire rulesets or individual rules, all thoroughly documented.

CodeAnalysis - IAR Embedded Workbench IDE - Arm 9	.60.3			_		×		
<u>File Edit View Project I-jet Tools Window H</u> e	lp							
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	- < Q > ⇆ म < 🖁 > 🗈 🗈 🛽 🕲 📼 🕻	A PARTY AND A P						
Workspace	startup_stm32f429x.s system_stm32f4xx.c main.c c_stat.c x	•••••				*		
	sancop_sinteressa system_sinteressa mante caute a							
CodeAnalysis - Debug ~	18					fo		
Files ♥ CodeAnalysis ● ● ● main.c ● ● ● Iar_stm32f429ij_aca.c ● ● ● stm32f4xx, i.c ● ● ● stm32f4xx, c ● ● ● StdPeriph_Driver ● ● ● Library.a ● ● ● stm32f4xx_gpio.c ● ● ● Library.a ● ● ● Library.a ● ● ● Library.a ● ● ● Library.bebug ✓	<pre>is void cert2(void) { static volatile int **ip; static volatile int **ip; static volatile int i = 0; volatile volatile int i = 0; volatile volatile int i = 0; volatile volatile volatile int int i int int</pre>	latile object ile reference been initialized,						
Overview CodeAnalysis Library	41					× a		
C-STAT Messages	L					•		
Severity: All 🗸 Filter: 🗸	Messages: 160							
Message		Check	Severity	File	Line	-		
A More than one arguments for the function A Conversion from a void pointer to an o B Oniter variable `p' is allocated but no	<pre>tf without detecting and handling errors string being consumed ated f without detecting and handling errors or casting explicitly to `void' tion call `c(a(), b())' are function calls themselves. The evaluation biject pointer t freed, returned or passed as an argument on all paths, causing a ax 100 which may be out of array bounds [0,15] 'h()'</pre>	CERT-EXP30-C_b CERT-EXP36-C_b	High	c_stat.c c_stat.c c_stat.c c_stat.c c_stat.c c_stat.c c_stat.c c_stat.c c_stat.c c_stat.c c_stat.c c_stat.c c_stat.c c_stat.c	3 3 3 3 5 5 6 6 6 6 6 6 6 6	36 36 36 36 37 37 30 55 30 55 30 51 31 37		
	-							
Build Debug Log C-STAT Messages Ready		Ln 37	, Col 26	System CAP				

IAR Embedded Workbench

IAR Embedded Workbench is a complete C/C++ development toolchain for embedded applications, offering high-quality code, optimized performance, and extensive debugging tools. The integration of C-STAT helps developers ensure code quality early, reducing errors and speeding up time to market.

Do you need to ensure your code is secure, high-quality, and standard-compliant?

We're here to support you, from project start to product life-cycle management.

Contact your local IAR team to get started: iar.com/contact!