Análise Estática de Código com Cppcheck

Contexto: Residência em Software Automotivo (CIN / Stellantis)

Professor Breno Miranda

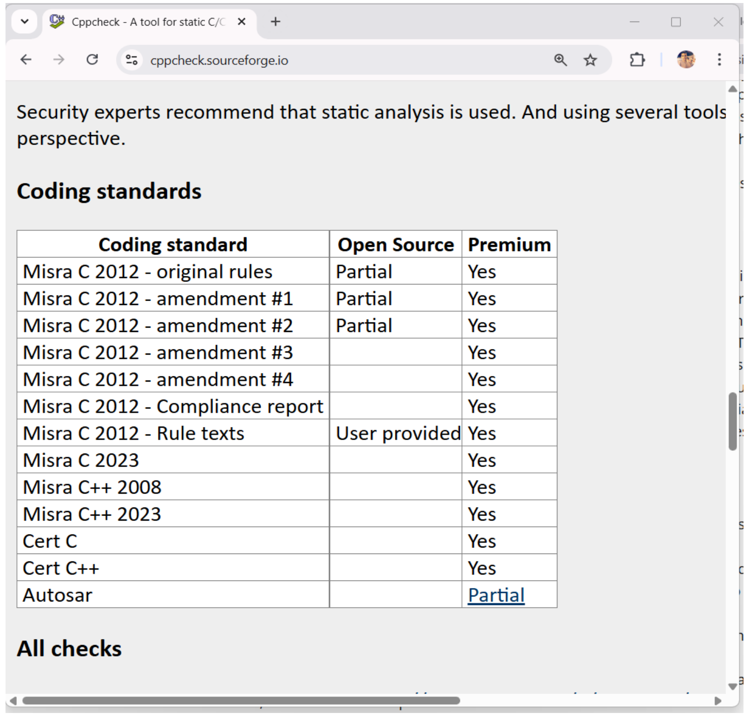
Autor Izaac Moraes de Oliveira

O Que é Análise Estática?

- **Definição:** Análise do código fonte **sem execução**.
- **Compilador vs. Analisador Estático:**
  - *Compilador:* Verifica Sintaxe (Gramática). "O código está escrito certo?"
  - *Análise Estática:* Verifica Semântica e Risco. "O código faz sentido ou vai quebrar?"
- **Destaque:**
  - 🚫 Não requer instrumentação (não altera o binário).
  - 🛡 Previne erros de tempo de execução (*Runtime Errors*).

Centro de Informática UFPE

STELLANTIS FINANCIAL SERVICES BR



Cppcheck - A tool for static C/C

cppcheck.sourceforge.io

Security experts recommend that static analysis is used. And using several tools perspective.

Coding standards

Coding standard	Open Source	Premium
Misra C 2012 - original rules	Partial	Yes
Misra C 2012 - amendment #1	Partial	Yes
Misra C 2012 - amendment #2	Partial	Yes
Misra C 2012 - amendment #3		Yes
Misra C 2012 - amendment #4		Yes
Misra C 2012 - Compliance report		Yes
Misra C 2012 - Rule texts	User provided	Yes
Misra C 2023		Yes
Misra C++ 2008		Yes
Misra C++ 2023		Yes
Cert C		Yes
Cert C++		Yes
Autosar		Partial

All checks

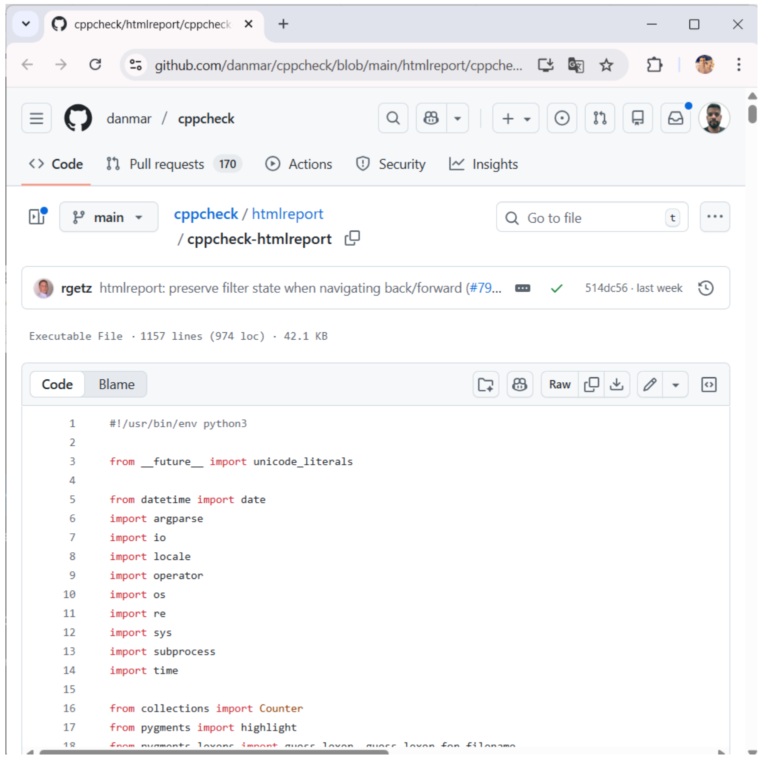
Por que Cppcheck? Diferenciais e Segurança

1. **Open Source & Comunidade:**
   - Licença GPL (Gratuito).
   - Projeto ativo no GitHub (`danmar/cppcheck`).
   - Documentação completa (Manual Online).
2. **Foco Técnico:**
   - Especialista em **Undefined Behaviour** (Comportamento Indefinido).
   - Detecta Dead Pointers, Integer Overflows, Divisão por Zero.
3. **Contexto Automotivo:**
   - Suporte a regras **MISRA C 2012**.
   - Suporte a regras **AUTOSAR**.
   - Versão Premium disponível para certificação ISO 26262 (Segurança Funcional).

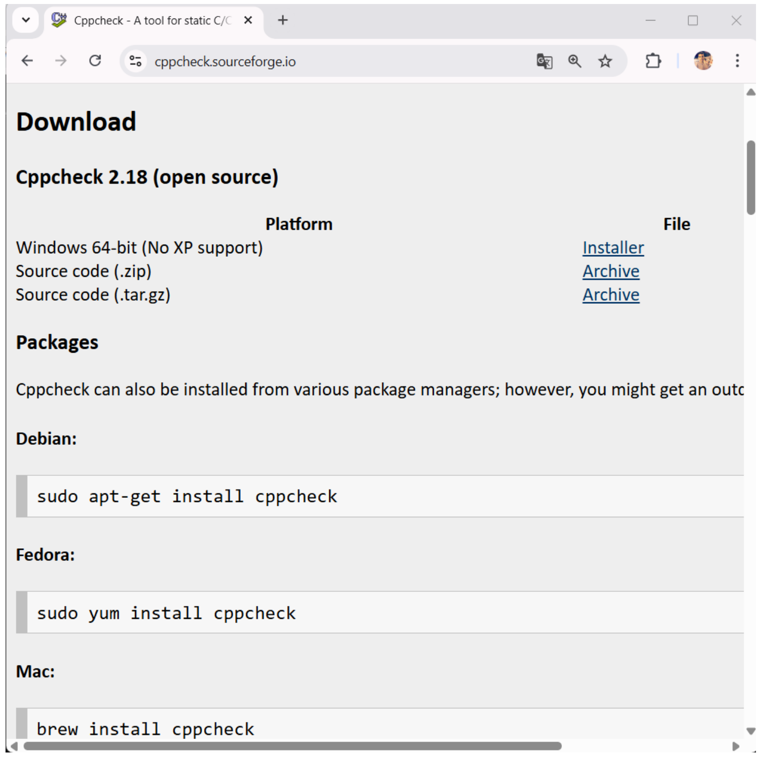
STELLANTIS FINANCIAL SERVICES BR — Centro de Informática UFPE

Instalação e Configuração: Pontos de Atenção

- **Nota:** Script `cppcheck-htmlreport.py`. - Necessário para converter XML em HTML."
- O script é mantido no repositório oficial do Cppcheck no GitHub: cppcheck-htmlreport no GitHub
- Basta baixá-lo do GitHub oficial, colocá-lo na raiz do projeto e garantir que a biblioteca `pygments` esteja instalada via pip.


cppcheck/htmlreport/cppcheck
github.com/danmar/cppcheck/blob/main/htmlreport/cppche...
danmar / cppcheck
Code
Pull requests 170
Actions
Security
Insights
main
cppcheck / htmlreport / cppcheck-htmlreport
Go to file
rgetz htmlreport: preserve filter state when navigating back/forward (#79...
514dc56 · last week
Executable File · 1157 lines (974 loc) · 42.1 KB
Code
Blame
Raw

```
#!/usr/bin/env python3

from __future__ import unicode_literals

from datetime import date
import argparse
import io
import locale
import operator
import os
import re
import sys
import subprocess
import time

from collections import Counter
from pygments import highlight
from pygments.lexers import guess_lexer, guess_lexer_for_filename
```



Download

Cppcheck 2.18 (open source)

Platform	File
Windows 64-bit (No XP support)	Installer
Source code (.zip)	Archive
Source code (.tar.gz)	Archive

Packages

Cppcheck can also be installed from various package managers; however, you might get an outc

Debian:

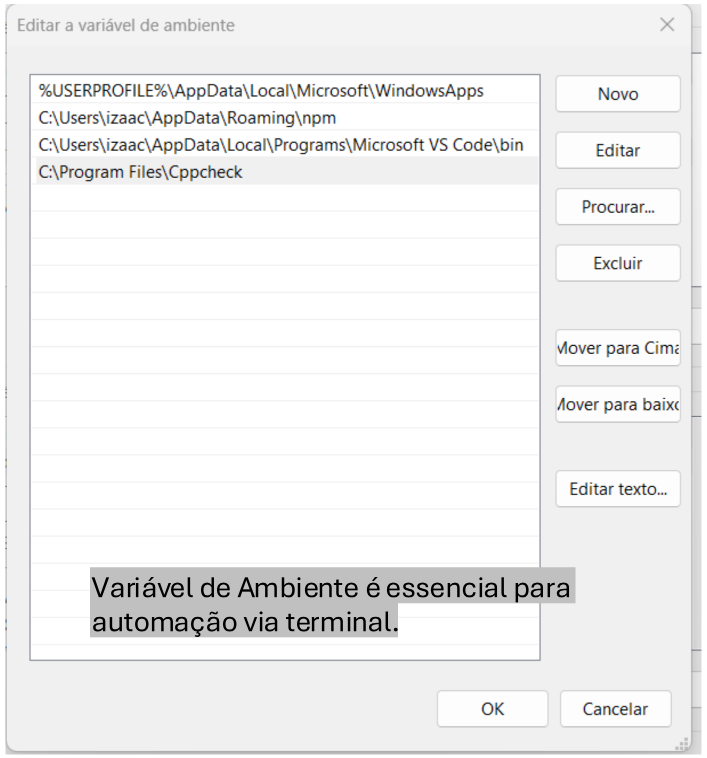
```
sudo apt-get install cppcheck
```

Fedora:

```
sudo yum install cppcheck
```

Mac:

```
brew install cppcheck
```



```
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Users\izaac\AppData\Roaming\npm
C:\Users\izaac\AppData\Local\Programs\Microsoft VS Code\bin
C:\Program Files\Cppcheck
```

Variável de Ambiente é essencial para automação via terminal.

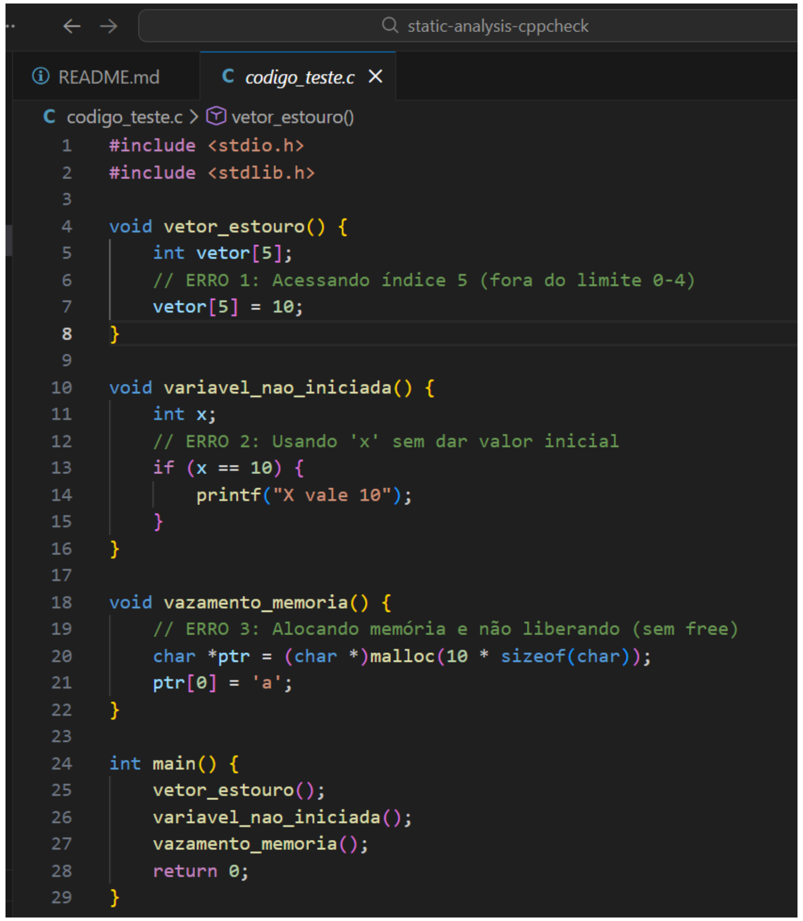
O Cenário de Teste: Código Vulnerável

Linha do Vetor: `vetor[5] = 10;` -> Etiqueta: *Buffer Overflow.*

Linha do IF: `if (x == 10)` -> Etiqueta: *Variável Não Inicializada.*

Linha do Malloc: `ptr[0] = 'a'` -> Etiqueta: *Risco de NULL Pointer.*

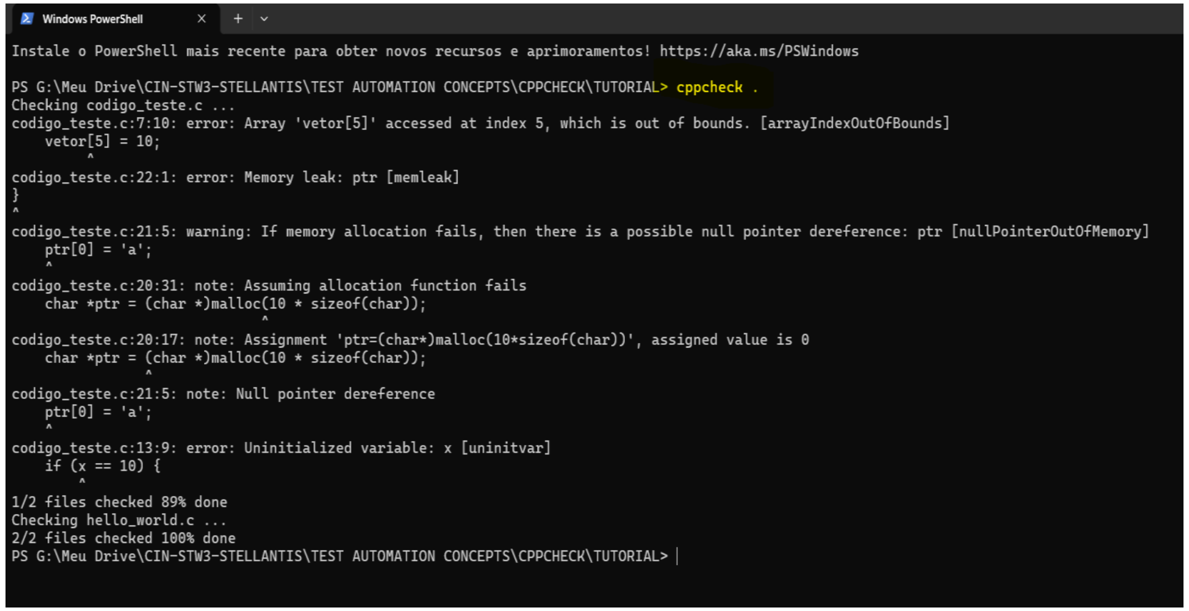
Fim da função: (Ausência do `free`) -> Etiqueta: *Memory Leak.*



```c
#include <stdio.h>
#include <stdlib.h>

void vetor_estouro() {
    int vetor[5];
    // ERRO 1: Acessando índice 5 (fora do limite 0-4)
    vetor[5] = 10;
}

void variavel_nao_iniciada() {
    int x;
    // ERRO 2: Usando 'x' sem dar valor inicial
    if (x == 10) {
        printf("X vale 10");
    }
}

void vazamento_memoria() {
    // ERRO 3: Alocando memória e não liberando (sem free)
    char *ptr = (char *)malloc(10 * sizeof(char));
    ptr[0] = 'a';
}

int main() {
    vetor_estouro();
    variavel_nao_iniciada();
    vazamento_memoria();
    return 0;
}
```

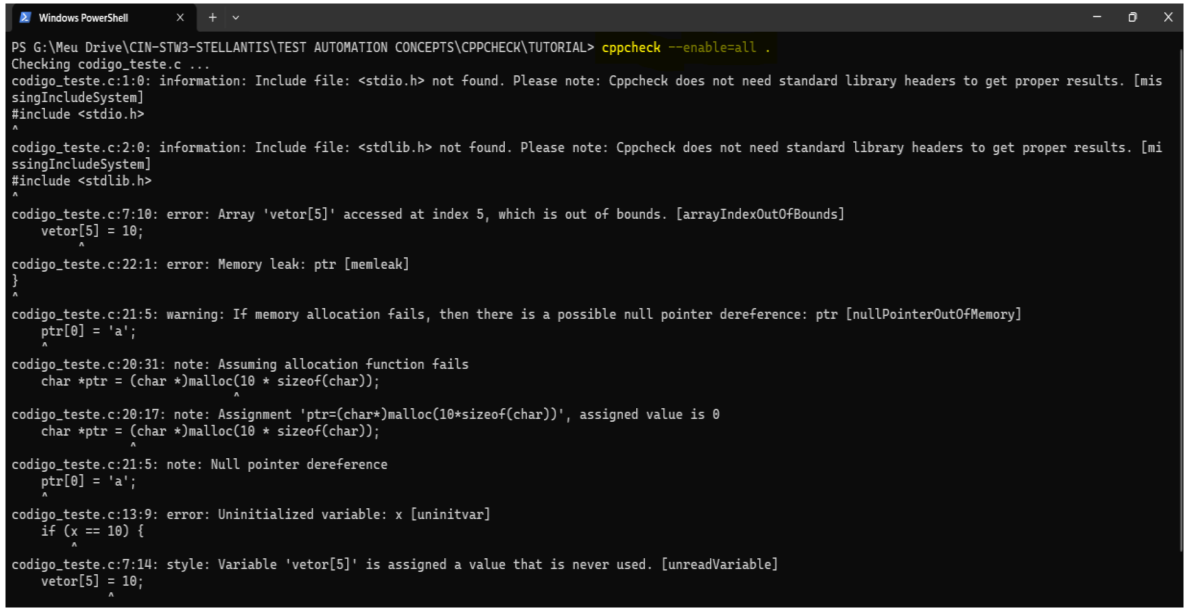
Execução, IDs e Modo Verboso

Varrendo todo diretório - Utilização de . / Flag –enable=all



```
Instale o PowerShell mais recente para obter novos recursos e aprimoramentos! https://aka.ms/PSWindows

PS G:\Meu Drive\CIN-STW3-STELLANTIS\TEST AUTOMATION CONCEPTS\CPPCHECK\TUTORIAL> cppcheck .
Checking codigo_teste.c ...
codigo_teste.c:7:10: error: Array 'vetor[5]' accessed at index 5, which is out of bounds. [arrayIndexOutOfBounds]
    vetor[5] = 10;
         ^
codigo_teste.c:22:1: error: Memory leak: ptr [memleak]
}
^
codigo_teste.c:21:5: warning: If memory allocation fails, then there is a possible null pointer dereference: ptr [nullPointerOutOfMemory]
    ptr[0] = 'a';
    ^
codigo_teste.c:20:31: note: Assuming allocation function fails
    char *ptr = (char *)malloc(10 * sizeof(char));
                              ^
codigo_teste.c:20:17: note: Assignment 'ptr=(char*)malloc(10*sizeof(char))', assigned value is 0
    char *ptr = (char *)malloc(10 * sizeof(char));
                ^
codigo_teste.c:21:5: note: Null pointer dereference
    ptr[0] = 'a';
    ^
codigo_teste.c:13:9: error: Uninitialized variable: x [uninitvar]
    if (x == 10) {
        ^
1/2 files checked 89% done
Checking hello_world.c ...
2/2 files checked 100% done
PS G:\Meu Drive\CIN-STW3-STELLANTIS\TEST AUTOMATION CONCEPTS\CPPCHECK\TUTORIAL> |
```

Busca apenas erros fatais

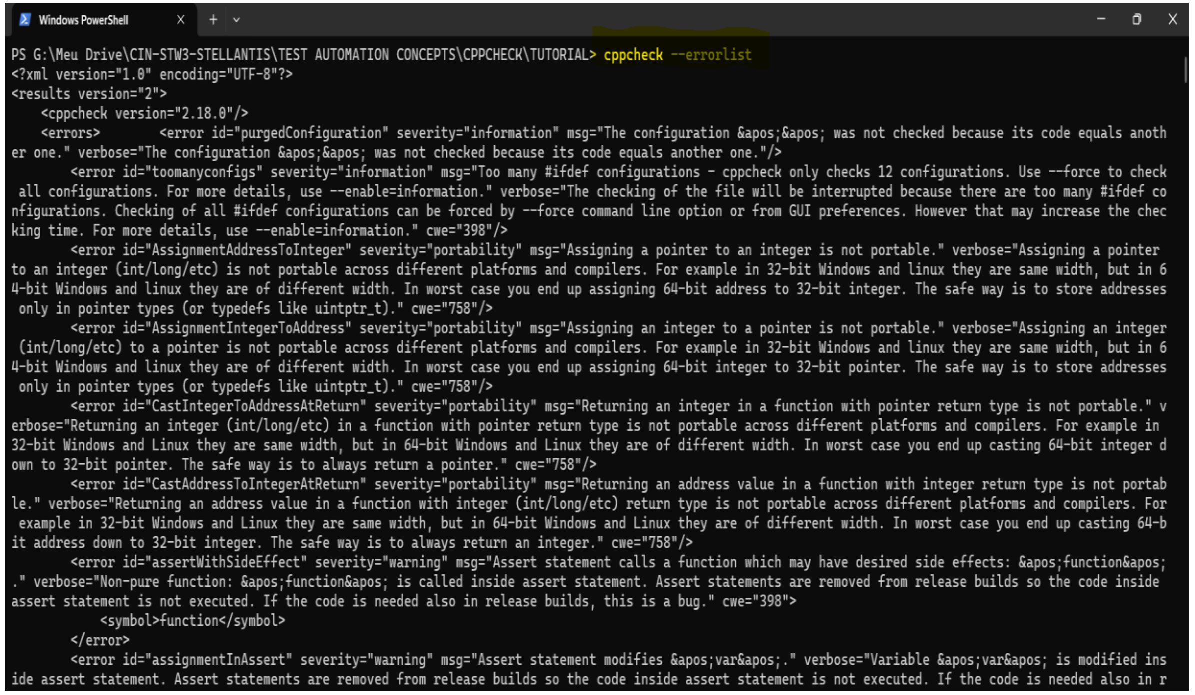


```
PS G:\Meu Drive\CIN-STW3-STELLANTIS\TEST AUTOMATION CONCEPTS\CPPCHECK\TUTORIAL> cppcheck --enable=all .
Checking codigo_teste.c ...
codigo_teste.c:1:0: information: Include file: <stdio.h> not found. Please note: Cppcheck does not need standard library headers to get proper results. [mis
singIncludeSystem]
#include <stdio.h>
^
codigo_teste.c:2:0: information: Include file: <stdlib.h> not found. Please note: Cppcheck does not need standard library headers to get proper results. [mi
ssingIncludeSystem]
#include <stdlib.h>
^
codigo_teste.c:7:10: error: Array 'vetor[5]' accessed at index 5, which is out of bounds. [arrayIndexOutOfBounds]
    vetor[5] = 10;
         ^
codigo_teste.c:22:1: error: Memory leak: ptr [memleak]
}
^
codigo_teste.c:21:5: warning: If memory allocation fails, then there is a possible null pointer dereference: ptr [nullPointerOutOfMemory]
    ptr[0] = 'a';
    ^
codigo_teste.c:20:31: note: Assuming allocation function fails
    char *ptr = (char *)malloc(10 * sizeof(char));
                              ^
codigo_teste.c:20:17: note: Assignment 'ptr=(char*)malloc(10*sizeof(char))', assigned value is 0
    char *ptr = (char *)malloc(10 * sizeof(char));
                ^
codigo_teste.c:21:5: note: Null pointer dereference
    ptr[0] = 'a';
    ^
codigo_teste.c:13:9: error: Uninitialized variable: x [uninitvar]
    if (x == 10) {
        ^
codigo_teste.c:7:14: style: Variable 'vetor[5]' is assigned a value that is never used. [unreadVariable]
    vetor[5] = 10;
             ^
```

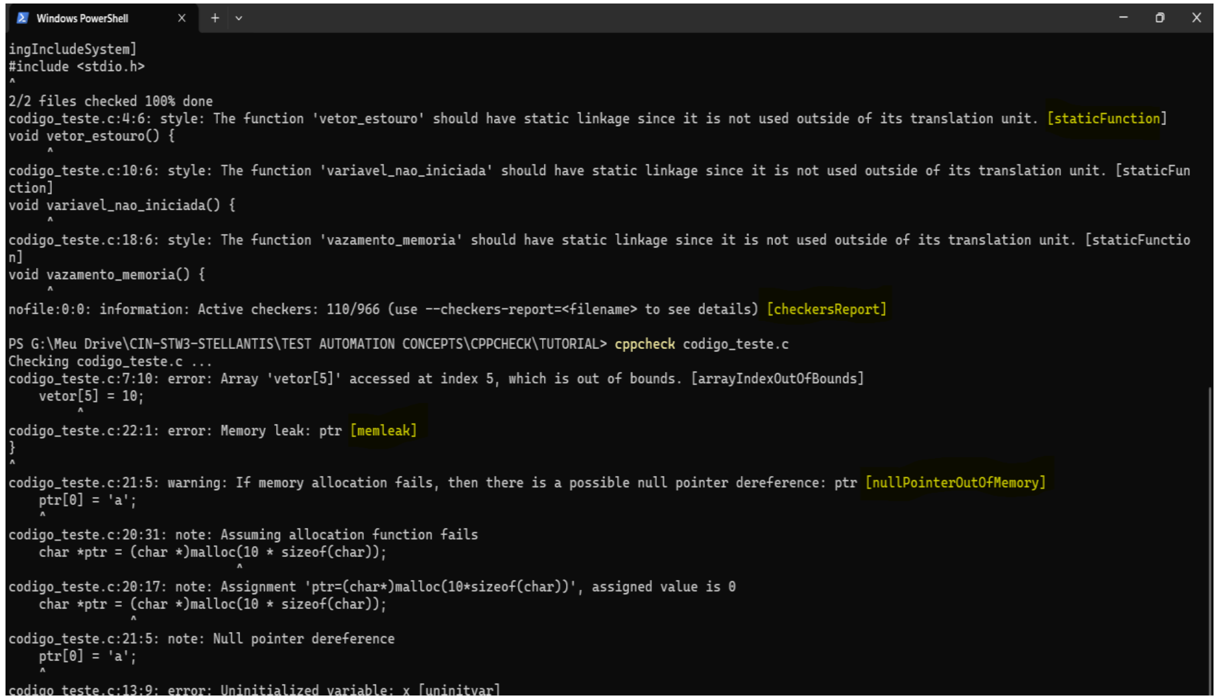
O ideal é sempre adicionar a flag **`--enable=all`**. Ela ativa verificadores de estilo, performance e portabilidade

Execução, IDs e Modo Verboso

Entendendo os IDs e cppcheck --errorlist



```
PS G:\Meu Drive\CIN-STW3-STELLANTIS\TEST AUTOMATION CONCEPTS\CPPCHECK\TUTORIAL> cppcheck --errorlist
<?xml version="1.0" encoding="UTF-8"?>
<results version="2">
    <cppcheck version="2.18.0"/>
    <errors>        <error id="purgedConfiguration" severity="information" msg="The configuration &apos;&apos; was not checked because its code equals anoth
er one." verbose="The configuration &apos;&apos; was not checked because its code equals another one."/>
        <error id="toomanyconfigs" severity="information" msg="Too many #ifdef configurations - cppcheck only checks 12 configurations. Use --force to check
 all configurations. For more details, use --enable=information." verbose="The checking of the file will be interrupted because there are too many #ifdef co
nfigurations. Checking of all #ifdef configurations can be forced by --force command line option or from GUI preferences. However that may increase the chec
king time. For more details, use --enable=information." cwe="398"/>
        <error id="AssignmentAddressToInteger" severity="portability" msg="Assigning a pointer to an integer is not portable." verbose="Assigning a pointer
to an integer (int/long/etc) is not portable across different platforms and compilers. For example in 32-bit Windows and linux they are same width, but in 6
4-bit Windows and linux they are of different width. In worst case you end up assigning 64-bit address to 32-bit integer. The safe way is to store addresses
 only in pointer types (or typedefs like uintptr_t)." cwe="758"/>
        <error id="AssignmentIntegerToAddress" severity="portability" msg="Assigning an integer to a pointer is not portable." verbose="Assigning an integer
 (int/long/etc) to a pointer is not portable across different platforms and compilers. For example in 32-bit Windows and linux they are same width, but in 6
4-bit Windows and linux they are of different width. In worst case you end up assigning 64-bit integer to 32-bit pointer. The safe way is to store addresses
 only in pointer types (or typedefs like uintptr_t)." cwe="758"/>
        <error id="CastIntegerToAddressAtReturn" severity="portability" msg="Returning an integer in a function with pointer return type is not portable." v
erbose="Returning an integer (int/long/etc) in a function with pointer return type is not portable across different platforms and compilers. For example in
32-bit Windows and Linux they are same width, but in 64-bit Windows and Linux they are of different width. In worst case you end up casting 64-bit integer d
own to 32-bit pointer. The safe way is to always return a pointer." cwe="758"/>
        <error id="CastAddressToIntegerAtReturn" severity="portability" msg="Returning an address value in a function with integer return type is not portab
le." verbose="Returning an address value in a function with integer (int/long/etc) return type is not portable across different platforms and compilers. For
 example in 32-bit Windows and Linux they are same width, but in 64-bit Windows and Linux they are of different width. In worst case you end up casting 64-b
it address down to 32-bit integer. The safe way is to always return an integer." cwe="758"/>
        <error id="assertWithSideEffect" severity="warning" msg="Assert statement calls a function which may have desired side effects: &apos;function&apos;
." verbose="Non-pure function: &apos;function&apos; is called inside assert statement. Assert statements are removed from release builds so the code inside
assert statement is not executed. If the code is needed also in release builds, this is a bug." cwe="398">
            <symbol>function</symbol>
        </error>
        <error id="assignmentInAssert" severity="warning" msg="Assert statement modifies &apos;var&apos;." verbose="Variable &apos;var&apos; is modified ins
ide assert statement. Assert statements are removed from release builds so the code inside assert statement is not executed. If the code is needed also in r
```



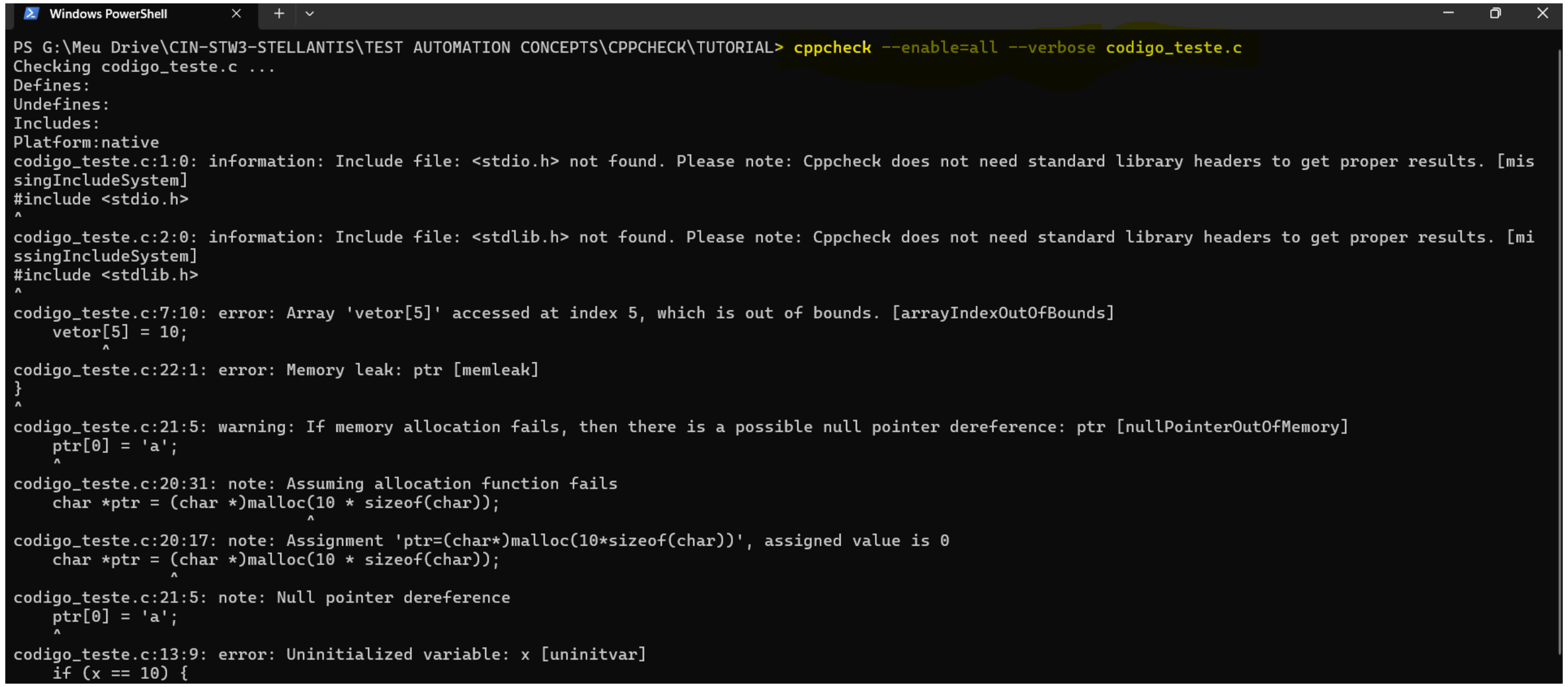
```
ingIncludeSystem]
#include <stdio.h>
^
2/2 files checked 100% done
codigo_teste.c:4:6: style: The function 'vetor_estouro' should have static linkage since it is not used outside of its translation unit. [staticFunction]
void vetor_estouro() {
     ^
codigo_teste.c:10:6: style: The function 'variavel_nao_iniciada' should have static linkage since it is not used outside of its translation unit. [staticFun
ction]
void variavel_nao_iniciada() {
     ^
codigo_teste.c:18:6: style: The function 'vazamento_memoria' should have static linkage since it is not used outside of its translation unit. [staticFunctio
n]
void vazamento_memoria() {
     ^
nofile:0:0: information: Active checkers: 110/966 (use --checkers-report=<filename> to see details) [checkersReport]

PS G:\Meu Drive\CIN-STW3-STELLANTIS\TEST AUTOMATION CONCEPTS\CPPCHECK\TUTORIAL> cppcheck codigo_teste.c
Checking codigo_teste.c ...
codigo_teste.c:7:10: error: Array 'vetor[5]' accessed at index 5, which is out of bounds. [arrayIndexOutOfBounds]
    vetor[5] = 10;
          ^
codigo_teste.c:22:1: error: Memory leak: ptr [memleak]
}
^
codigo_teste.c:21:5: warning: If memory allocation fails, then there is a possible null pointer dereference: ptr [nullPointerOutOfMemory]
    ptr[0] = 'a';
    ^
codigo_teste.c:20:31: note: Assuming allocation function fails
    char *ptr = (char *)malloc(10 * sizeof(char));
                              ^
codigo_teste.c:20:17: note: Assignment 'ptr=(char*)malloc(10*sizeof(char))', assigned value is 0
    char *ptr = (char *)malloc(10 * sizeof(char));
                ^
codigo_teste.c:21:5: note: Null pointer dereference
    ptr[0] = 'a';
    ^
codigo_teste.c:13:9: error: Uninitialized variable: x [uninitvar]
```

Para ver todos os erros que a ferramenta é capaz de detectar, utiliza-se o comando `cppcheck --errorlist`

ID entre colchetes, como `[memleak]` - o ID é a assinatura única do erro

Execução, IDs e Modo Verboso

flag **--verbose**



Windows PowerShell

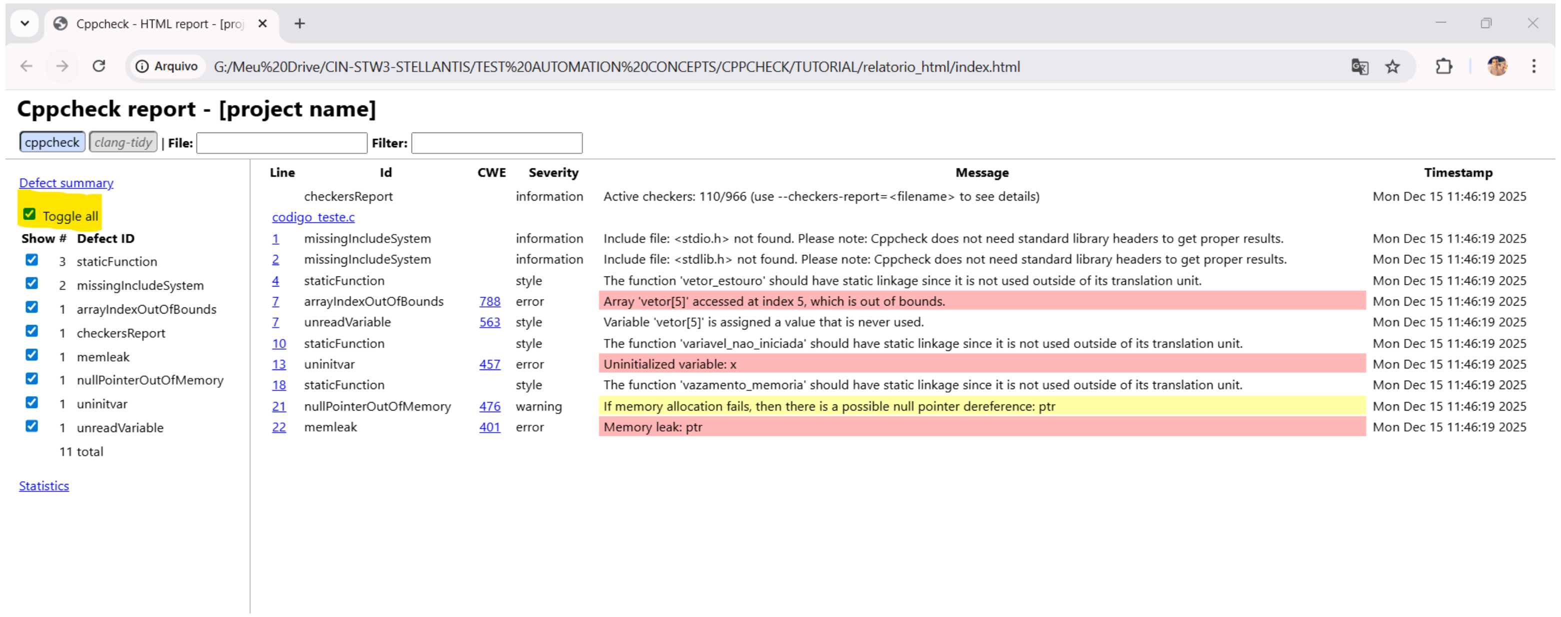
```
PS G:\Meu Drive\CIN-STW3-STELLANTIS\TEST AUTOMATION CONCEPTS\CPPCHECK\TUTORIAL> cppcheck --enable=all --verbose codigo_teste.c
Checking codigo_teste.c ...
Defines:
Undefines:
Includes:
Platform:native
codigo_teste.c:1:0: information: Include file: <stdio.h> not found. Please note: Cppcheck does not need standard library headers to get proper results. [mis
singIncludeSystem]
#include <stdio.h>
^
codigo_teste.c:2:0: information: Include file: <stdlib.h> not found. Please note: Cppcheck does not need standard library headers to get proper results. [mi
ssingIncludeSystem]
#include <stdlib.h>
^
codigo_teste.c:7:10: error: Array 'vetor[5]' accessed at index 5, which is out of bounds. [arrayIndexOutOfBounds]
    vetor[5] = 10;
         ^
codigo_teste.c:22:1: error: Memory leak: ptr [memleak]
}
^
codigo_teste.c:21:5: warning: If memory allocation fails, then there is a possible null pointer dereference: ptr [nullPointerOutOfMemory]
    ptr[0] = 'a';
    ^
codigo_teste.c:20:31: note: Assuming allocation function fails
    char *ptr = (char *)malloc(10 * sizeof(char));
                              ^
codigo_teste.c:20:17: note: Assignment 'ptr=(char*)malloc(10*sizeof(char))', assigned value is 0
    char *ptr = (char *)malloc(10 * sizeof(char));
                ^
codigo_teste.c:21:5: note: Null pointer dereference
    ptr[0] = 'a';
    ^
codigo_teste.c:13:9: error: Uninitialized variable: x [uninitvar]
    if (x == 10) {
```

No modo verboso, o Cppcheck conta a história do erro: ele mostra onde a memória foi alocada e onde ela vazou. Isso economiza minutos preciosos de debug.

Gerando Relatórios em XML & HTML

Gerar o XML: `cppcheck --xml --enable=all codigo_teste.c 2> relatorio.xml`

Converter para HTML: `python cppcheck-htmlreport.py --file=relatorio.xml --report-dir=relatorio_html --source-dir=.`


Cppcheck - HTML report - [proj
Arquivo
G:/Meu%20Drive/CIN-STW3-STELLANTIS/TEST%20AUTOMATION%20CONCEPTS/CPPCHECK/TUTORIAL/relatorio_html/index.html

Cppcheck report - [project name]

cppcheck | clang-tidy | **File:** | **Filter:**

Defect summary

✓ Toggle all

Show	#	Defect ID
✓	3	staticFunction
✓	2	missingIncludeSystem
✓	1	arrayIndexOutOfBounds
✓	1	checkersReport
✓	1	memleak
✓	1	nullPointerOutOfMemory
✓	1	uninitvar
✓	1	unreadVariable
	11	total

Statistics

Line	Id	CWE	Severity	Message	Timestamp
	checkersReport		information	Active checkers: 110/966 (use --checkers-report=<filename> to see details)	Mon Dec 15 11:46:19 2025
codigo_teste.c					
1	missingIncludeSystem		information	Include file: <stdio.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	Mon Dec 15 11:46:19 2025
2	missingIncludeSystem		information	Include file: <stdlib.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	Mon Dec 15 11:46:19 2025
4	staticFunction		style	The function 'vetor_estouro' should have static linkage since it is not used outside of its translation unit.	Mon Dec 15 11:46:19 2025
7	arrayIndexOutOfBounds	788	error	Array 'vetor[5]' accessed at index 5, which is out of bounds.	Mon Dec 15 11:46:19 2025
7	unreadVariable	563	style	Variable 'vetor[5]' is assigned a value that is never used.	Mon Dec 15 11:46:19 2025
10	staticFunction		style	The function 'variavel_nao_iniciada' should have static linkage since it is not used outside of its translation unit.	Mon Dec 15 11:46:19 2025
13	uninitvar	457	error	Uninitialized variable: x	Mon Dec 15 11:46:19 2025
18	staticFunction		style	The function 'vazamento_memoria' should have static linkage since it is not used outside of its translation unit.	Mon Dec 15 11:46:19 2025
21	nullPointerOutOfMemory	476	warning	If memory allocation fails, then there is a possible null pointer dereference: ptr	Mon Dec 15 11:46:19 2025
22	memleak	401	error	Memory leak: ptr	Mon Dec 15 11:46:19 2025

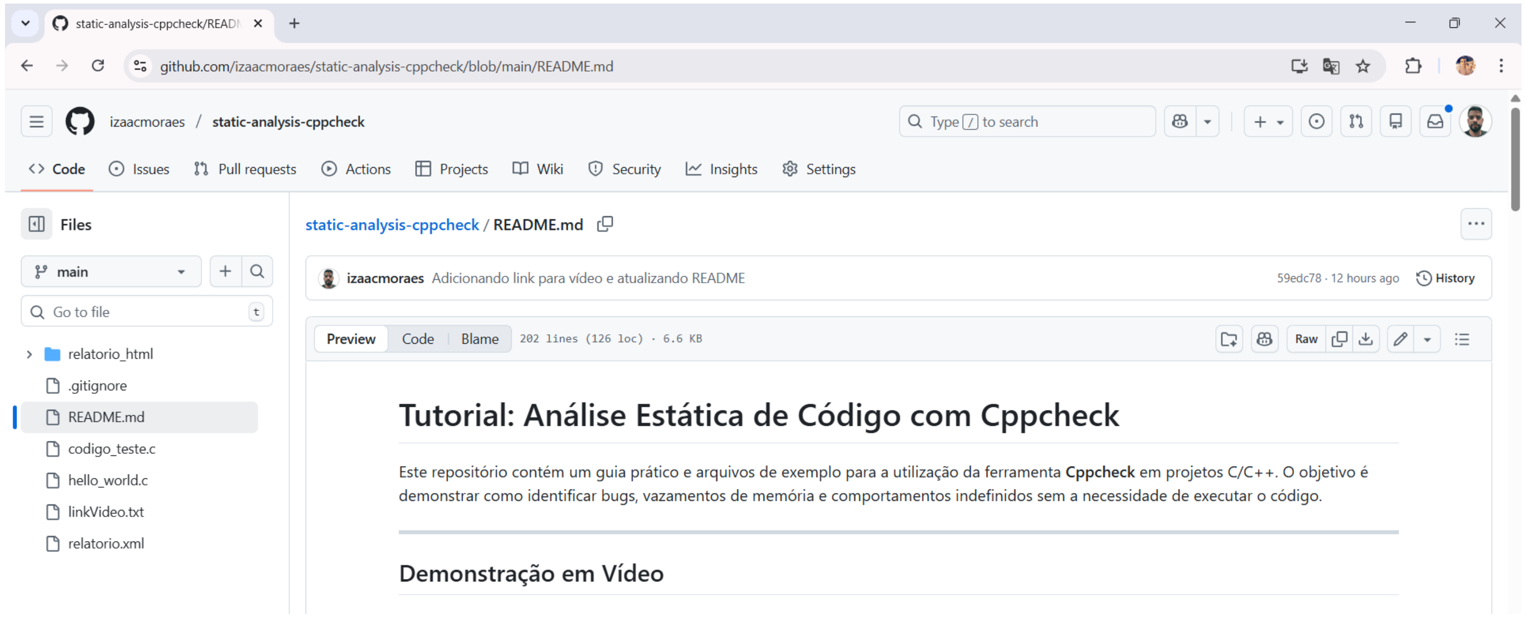
- flag `--xml` redirecionando a saída. Isso cria um arquivo estruturado com todos os dados brutos da análise.
- cript Python oficial `cppcheck-htmlreport`. Um detalhe importante aqui é o parâmetro `--source-dir=.`, que garante que o código fonte original seja copiado para o relatório
- a leitura fica intuitiva. Temos a lista de erros agrupada e, ao clicar, ele destaca em **vermelho** a linha exata do problema no código. Isso elimina a necessidade de abrir a IDE para entender o contexto do bug.

STELLANTIS FINANCIAL SERVICES BR

Centro de Informática UFPE

Gerando Relatórios em XML & HTML

- https://github.com/izaacmoraes/static-analysis-cppcheck


static-analysis-cppcheck/README
github.com/izaacmoraes/static-analysis-cppcheck/blob/main/README.md
izaacmoraes / static-analysis-cppcheck
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Files
main
Go to file
relatorio_html
.gitignore
README.md
codigo_teste.c
hello_world.c
linkVideo.txt
relatorio.xml
static-analysis-cppcheck / README.md
izaacmoraes Adicionando link para vídeo e atualizando README
59edc78 · 12 hours ago
History
Preview
Code
Blame
202 lines (126 loc) · 6.6 KB
Raw

Tutorial: Análise Estática de Código com Cppcheck

Este repositório contém um guia prático e arquivos de exemplo para a utilização da ferramenta **Cppcheck** em projetos C/C++. O objetivo é demonstrar como identificar bugs, vazamentos de memória e comportamentos indefinidos sem a necessidade de executar o código.

Demonstração em Vídeo