Name: Jake Gendreau

Date: December 6, 2024

**CYB220 Homework 5– Static analysis tools**

**Objective:** Practice using static analysis tools to analyze programs.

**Due:** Friday, Dec 6, 2024 11:59pm on CANVAS.

**Turn in:** This report

Points: 60 points

**Requirements:**

* Take a look at the 5 vulnerable programs (3 C++, 2 C code), and understand the injected vulnerabilities.
  + Programs posted on CANVAS for this homework, *HW5-VulCode.zip*.
* Run 3 different static analysis tools to analyze the posted five vulnerable programs.
* While running the tools, finish the following report.

(15 pts) Step 1: Run 3 different static analysis tools to analyze the five vulnerable programs (posted on CANVAS for this homework, HW5-VulCode.zip).

The 3 static analysis tools are:

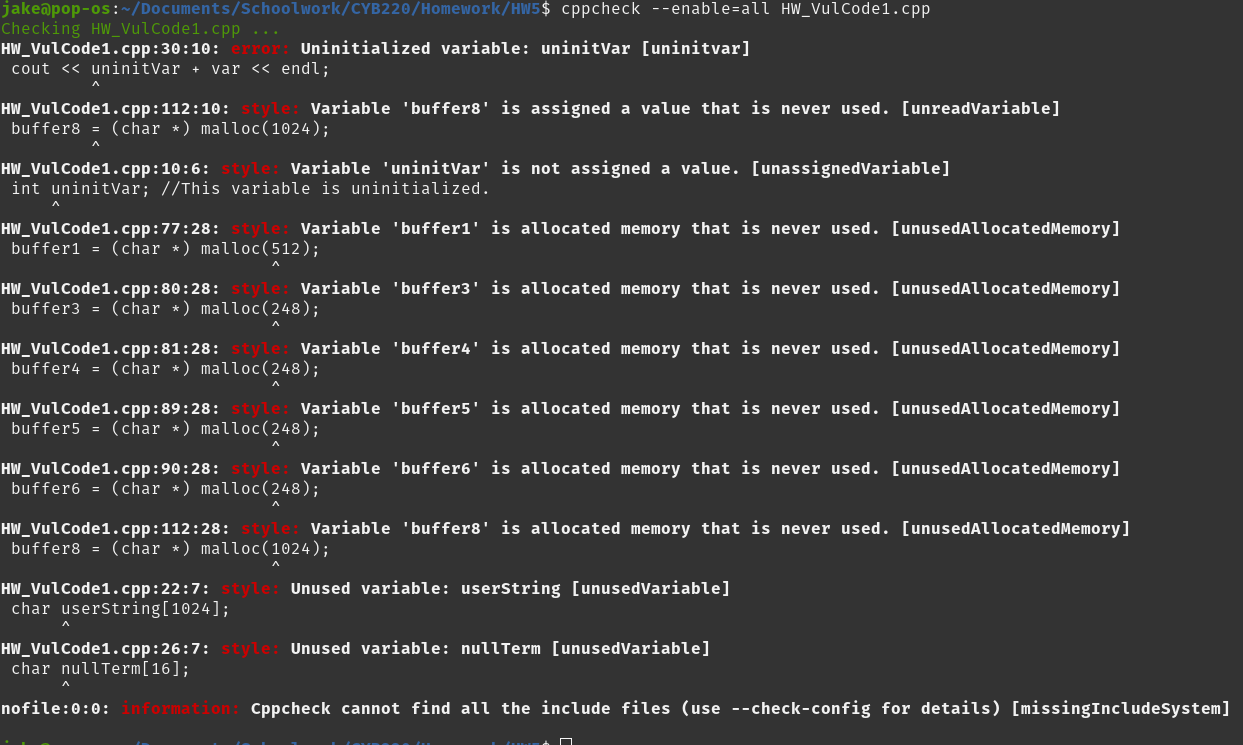
Clang/clang++ static analyzer ([https://clang-analyzer.llvm.org](https://clang-analyzer.llvm.org/))

Gcc/g++ static analyzer (<https://gcc.gnu.org/onlinedocs//gcc/Static-Analyzer-Options.html>)

Cppcheck ([https://cppcheck.sourceforge.io](https://cppcheck.sourceforge.io/))

# STATIC ANALYSIS TOOLs #1

(15 pts) Run static analysis tool #1 cppcheck to analyze the vulnerable programs HW\_VulCode1.cpp. The result from the tool is (take a screenshot of the results ) (A screenshot of analyzing one program is good enough; I don’t need five screenshots):

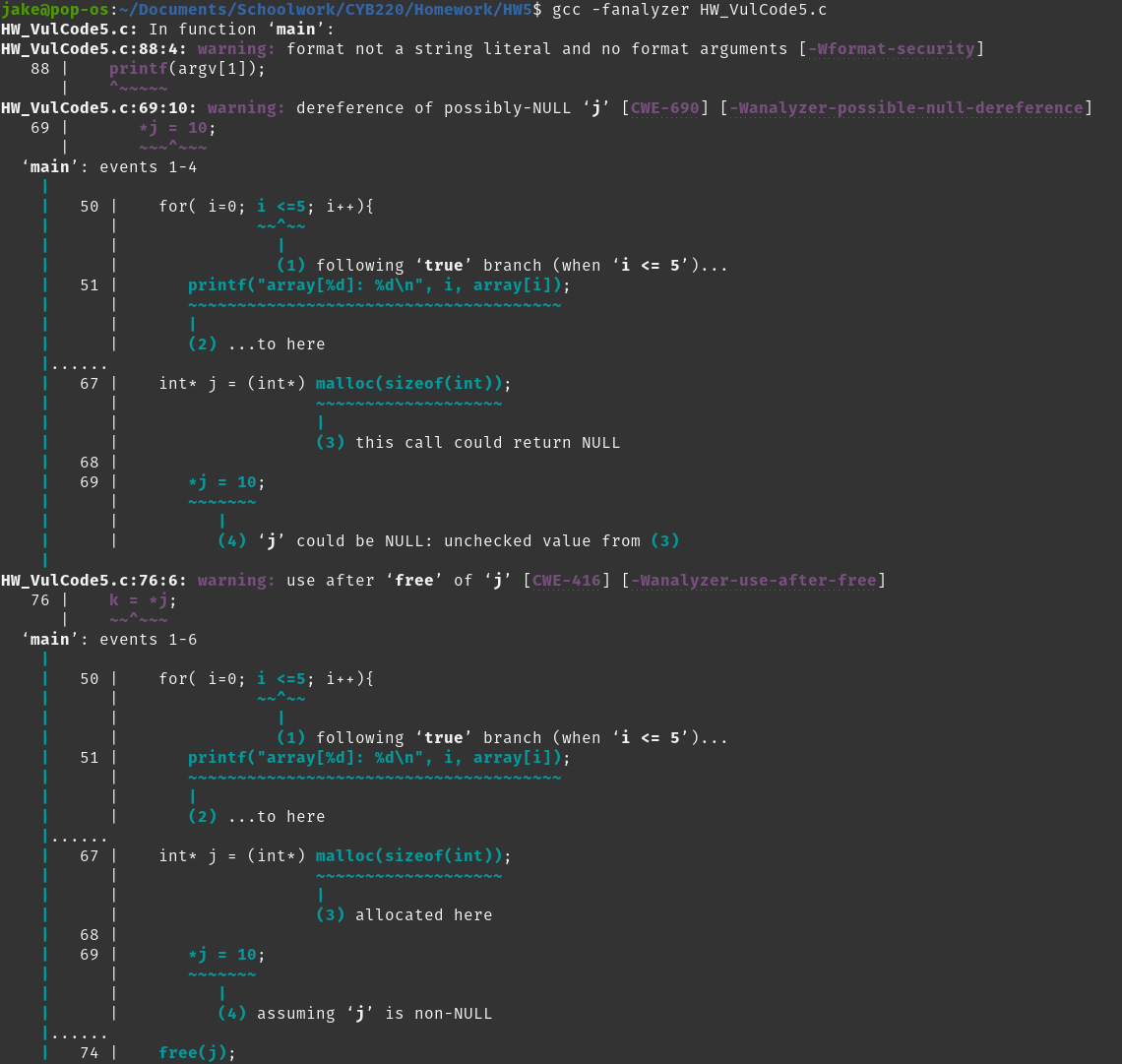


Question: What is the tool good for? Not good for? Does it give you false positive results?

cppcheck is really good for identifying errors rapidly and figuring out what is wrong with them. It does not offer solutions, but it covers the majority of common errors quickly and efficiently.

# g++ static analyzer

(15 pts) Run static analysis tool #2 g++ Static Analyser to analyze the vulnerable program HW\_VulCode5.c. The result from the tool is (take a screenshot):

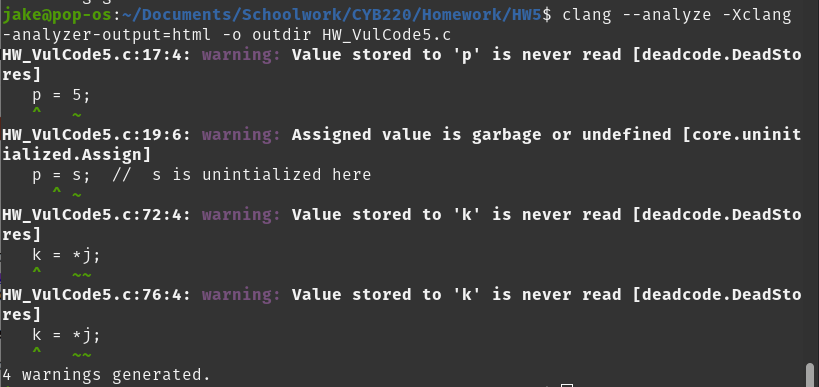


Question: What is the tool good for? Not good for? Does it give you false positive results?

The g++ compiler is good for identifying some issues and offering possible solutions. Because it offers solutions, it is not dense, and is harder to parse than cppcheck. It wasn't very thorough, and its only real advantage was that it was built=in. It also gave me lots of false positives regarding malloc() and its possibility to return NULL.

# Clang static analysis tool

(15 pts) Run static analysis tool #3 Clang Static Analysis tool to analyze the vulnerable program HW\_VulCode5.c. The result from the tool is:



Question: What is the tool good for? Not good for? Does it give you false positive results?

I despise this tool. It requires discrete installation, is not easy to use, and came up with minimal results. I couldn't find anywhere that simply tells how to use the tool, and it kept bringing up " iostream file not found", which stopped error checking then and there. I don't know how to fix this, and I feel that the tool should be better than that. cppcheck certainly is.

**Summary**

(15 pts) Step 2: Final summary of using the 3 static analysis tools. Compare these 3 static analysis tools. For columns 2, 3 and 4, if the tool can identify the bug, mark “YES”. Otherwise, mark “NO”.

|  |  |  |  |
| --- | --- | --- | --- |
| Vulnerability | cppcheck | g++ static check | (Tool #3 name) |
| Uninitialized variable | YES | NO | NO |
| Buffer overflow | YES | NO | YES |
| Integer overflow | YES | NO | NO |
| Off-by-one | YES | NO | NO |
| Type mismatch | NO | YES | NO |
| Use-after-free | YES | YES | NO |
| Double free | YES | YES | NO |
| Use of unsafe string functions | YES | YES | YES |
| Format string vulnerability | NO | YES | NO |
| Summary of the tool | This tool is efficient and catches the majority of common errors. | This tool takes a lot of room but doesn't cover a lot of results. | Not only is the documentation confusing, the tool is also not very useful. |