

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Practical-1 Static code analysis using open source tool Flawfinder

Date:-12/07/2024 Submission Date:- 19/07/2024

**Write-up:-**

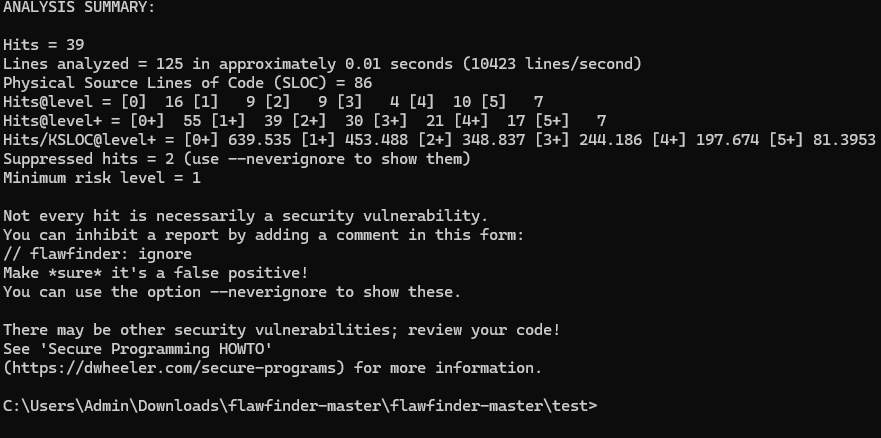
* Static code Analysis & Benefits
* Vulnerability
* Flawfinder

Implement static code analysis using open source tool Flawfinder for the following:

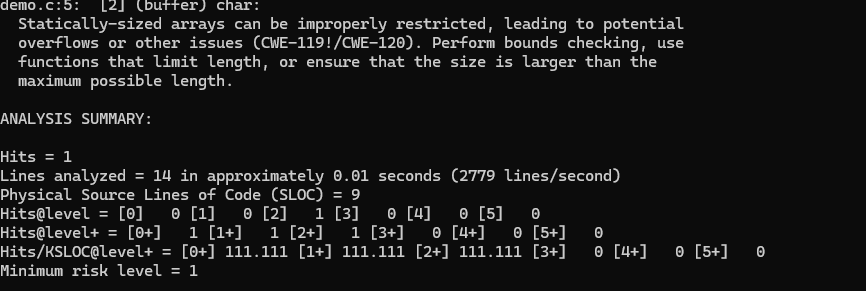
* Buffer overflow
* String problem
* Race conditions,etc

Steps taken to perform the practical:

1. Download Flawfinder from Github
2. Using “pip install flawfinder” Install flawfinder using command prompt
3. Once flawfinder is installed we will test the given ‘test.c’ file given in the ‘test’ folder in the flawfinder folder.
4. Open command prompt and type “flawfinder test.c”
5. It will run and give the following hits



1. Created C Program to get array error



1. Created C Program to get race condition error:

Race Condition prac 1 paper 2

int main() {

    char \* fn = "/tmp/XYZ";

    char buffer[60];

    FILE \*fp;

    /\* get user input \*/

    scanf("%50s", buffer );

    if(!access(fn, W\_OK)){

        fp = fopen(fn, "a+");

        fwrite("\n", sizeof(char), 1, fp);

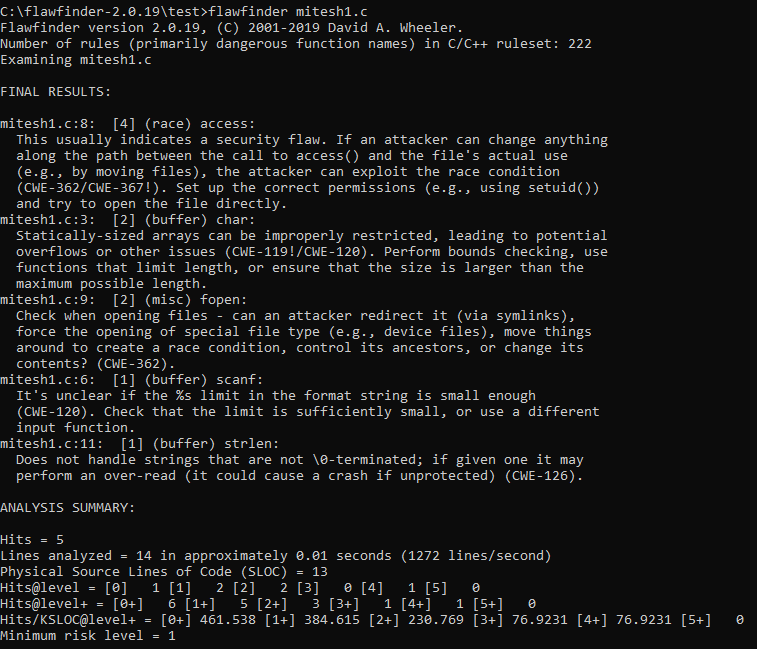
        fwrite(buffer, sizeof(char), strlen(buffer), fp);

        fclose(fp);

    }

    else printf("No permission \n");

}



1. Created program to create buffer overflow error

// A C program to demonstrate buffer overflow

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main(int argc, char \*argv[])

{

    // Reserve 5 byte of buffer plus the terminating NULL.

    // should allocate 8 bytes = 2 double words,

    // To overflow, need more than 8 bytes...

    char buffer[5]; // If more than 8 characters input

                        // by user, there will be access

                        // violation, segmentation fault

    // a prompt how to execute the program...

    if (argc < 2)

    {

            printf("strcpy() NOT executed....\n");

            printf("Syntax: %s <characters>\n", argv[0]);

            exit(0);

    }

    // copy the user input to mybuffer, without any

    // bound checking a secure version is strcpy\_s()

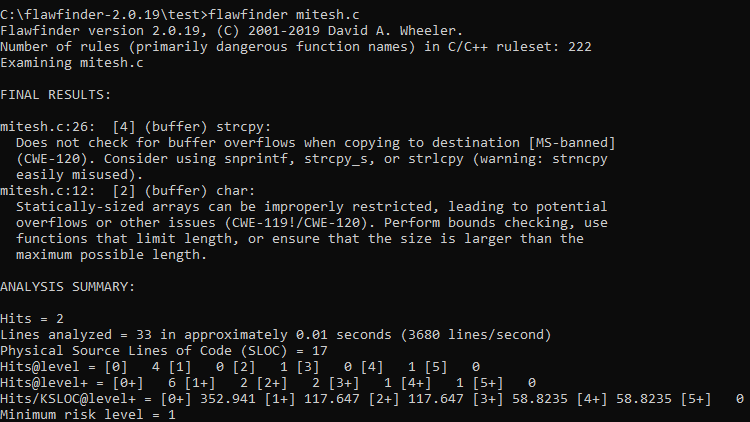
    strcpy(buffer, argv[1]);

    printf("buffer content= %s\n", buffer);

    // you may want to try strcpy\_s()

    printf("strcpy() executed...\n");

    return 0;



1. Created program to for string error

#include <stdio.h>

#include <string.h>

int main() {

    char buffer[5]; // Buffer to store strings (size 4 + 1 for null terminator)

    char \*fruits[] = {"apple", "banana", "mango", "carrot", "chikoo", "jackfruit"};

    int i;

    for (i = 0; i < sizeof(fruits) / sizeof(fruits[0]); i++) {

        strncpy(buffer, fruits[i], sizeof(buffer) - 1);

        buffer[sizeof(buffer) - 1] = '\0'; // Ensure null-termination

        printf("String %d: %s\n", i + 1, buffer);

        // If the string is longer than the buffer, handle accordingly

        if (strlen(fruits[i]) > sizeof(buffer) - 1) {

            printf("Warning: String '%s' truncated to fit buffer size.\n", fruits[i]);

        }

    }

    return 0;

}

