

**Software Security A.A. 2021-2022**  
**Individual Project 1**  
**Stanco Donato Francesco Pio 2027523**

## 1 – Up to 3 lines stating the major strengths and weaknesses of the tool

Il principale punto di forza del tool Flawfinder è la sua semplicità di utilizzo.

Mentre, la debolezza che risalta è che usa un semplice parsing basato su token, quindi non tutti gli hit segnalati sono necessariamente vulnerabilità di sicurezza e alcune non vengono neanche individuate.

## 2 – A screenshot of the output of the tool for the fragment

```
donatostanco@MacBook-Pro-di-Donato ~ % flawfinder /Users/donatostanco/Desktop/Proj1_fa21/project1_FA21.c
Flawfinder version 2.0.19, (C) 2001-2019 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 222
Examining /Users/donatostanco/Desktop/Proj1_fa21/project1_FA21.c

FINAL RESULTS:

/Users/donatostanco/Desktop/Proj1_fa21/project1_FA21.c:42: [4] (buffer) strcpy:
Does not check for buffer overflows when copying to destination [MS-banned]
(CWE-120). Consider using snprintf, strcpy_s, or strncpy (warning: strncpy
easily misused).
/Users/donatostanco/Desktop/Proj1_fa21/project1_FA21.c:56: [4] (format) fprintf:
If format strings can be influenced by an attacker, they can be exploited
(CWE-134). Use a constant for the format specification.
/Users/donatostanco/Desktop/Proj1_fa21/project1_FA21.c:8: [2] (buffer) char:
Statically-sized arrays can be improperly restricted, leading to potential
overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
functions that limit length, or ensure that the size is larger than the
maximum possible length.
/Users/donatostanco/Desktop/Proj1_fa21/project1_FA21.c:28: [2] (buffer) char:
Statically-sized arrays can be improperly restricted, leading to potential
overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
functions that limit length, or ensure that the size is larger than the
maximum possible length.
/Users/donatostanco/Desktop/Proj1_fa21/project1_FA21.c:33: [2] (buffer) char:
Statically-sized arrays can be improperly restricted, leading to potential
overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
functions that limit length, or ensure that the size is larger than the
maximum possible length.
/Users/donatostanco/Desktop/Proj1_fa21/project1_FA21.c:35: [2] (buffer) strcat:
Does not check for buffer overflows when concatenating to destination
[MS-banned] (CWE-120). Consider using strcat_s, strncat, strlcat, or
snprintf (warning: strncat is easily misused). Risk is low because the
source is a constant string.
/Users/donatostanco/Desktop/Proj1_fa21/project1_FA21.c:8: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may
perform an over-read (it could cause a crash if unprotected) (CWE-126).
/Users/donatostanco/Desktop/Proj1_fa21/project1_FA21.c:9: [1] (buffer) strncpy:
Easily used incorrectly; doesn't always \0-terminate or check for invalid
pointers [MS-banned] (CWE-120).
/Users/donatostanco/Desktop/Proj1_fa21/project1_FA21.c:9: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may
perform an over-read (it could cause a crash if unprotected) (CWE-126).
/Users/donatostanco/Desktop/Proj1_fa21/project1_FA21.c:10: [1] (buffer) strlen:
Does not handle strings that are not \0-terminated; if given one it may
perform an over-read (it could cause a crash if unprotected) (CWE-126).
/Users/donatostanco/Desktop/Proj1_fa21/project1_FA21.c:17: [1] (buffer) read:
Check buffer boundaries if used in a loop including recursive loops
(CWE-120, CWE-20).
/Users/donatostanco/Desktop/Proj1_fa21/project1_FA21.c:22: [1] (buffer) read:
Check buffer boundaries if used in a loop including recursive loops
(CWE-120, CWE-20).
/Users/donatostanco/Desktop/Proj1_fa21/project1_FA21.c:34: [1] (buffer) strncpy:
Easily used incorrectly; doesn't always \0-terminate or check for invalid
pointers [MS-banned] (CWE-120).

ANALYSIS SUMMARY:

Hits = 13
Lines analyzed = 66 in approximately 0.01 seconds (6130 lines/second)
Physical Source Lines of Code (SLOC) = 53
Hits@level = [0] 2 [1] 7 [2] 4 [3] 0 [4] 2 [5] 0
Hits@level+ = [0+] 15 [1+] 13 [2+] 6 [3+] 2 [4+] 2 [5+] 0
Hits/KSLOC@level+ = [0+] 283.019 [1+] 245.283 [2+] 113.208 [3+] 37.7358 [4+] 37.7358 [5+] 0
Minimum risk level = 1

Not every hit is necessarily a security vulnerability.
You can inhibit a report by adding a comment in this form:
// flawfinder: ignore
Make *sure* it's a false positive!
You can use the option --neverignore to show these.

There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(https://dwheeler.com/secure-programs) for more information.
```

### 3 – An analysis of each warning

- *Hint 1: 42: [4] (buffer) strcpy: Does not check for buffer overflows when copying to destination [MS-banned] (CWE-120). Consider using snprintf, strcpy\_s, or strncpy (warning: strncpy easily misused).*

**Analisi:** si può verificare un buffer overflow se `foo` è più grande di `buffer`, per risolvere il problema potrebbe essere utilizzata la funzione `strncpy(buffer, foo, sizeof(buffer) - 1)` e poi si aggiunge il carattere di fine stringa `'\\0'`.

- *Hint 2: 56: [4] (format) fprintf: If format strings can be influenced by an attacker, they can be exploited (CWE-134). Use a constant for the format specification.*

**Analisi:** questo tipo di chiamata può provocare un buffer overflow. Per correggerlo si può usare il carattere `"%s"` (Format String Parameter) nella funzione `fprintf(stderr, "%s", message)`.

- *Hint 3: 8: [2] (buffer) char: Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.*

**Analisi:** falso positivo, non è possibile che avvenga un buffer overflow in quanto la dimensione di `dst` viene calcolata in base alla lunghezza di `src`, viene considerato anche il carattere di fine stringa.

- *Hint 4: 28: [2] (buffer) char: Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.*

**Analisi:** falso positivo, non è possibile che avvenga un buffer overflow, poiché in questo caso la funzione `fgets()` può al massimo inserire 1024 carattere in `buffer-1` considerando il fine stringa.

- *Hint 5: 33: [2] (buffer) char: Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use functions that limit length, or ensure that the size is larger than the maximum possible length.*

**Analisi:** falso positivo, non è possibile che avvenga un buffer overflow, poiché `errmsg` è grande abbastanza da contenere l'array `buffer` più la stringa nella funzione `strcat(errormsg, " is not a valid ID")`.

- *Hint 6: 35: [2] (buffer) strcat: Does not check for buffer overflows when concatenating to destination [MS-banned] (CWE-120). Consider using strcat\_s, strncat, strlcat, or snprintf (warning: strncat is easily misused). Risk is low because the source is a constant string.*

**Analisi:** falso positivo, non è possibile che avvenga un buffer overflow, in quanto `errmsg` è abbastanza grande da contenere l'array `buffer` e la stringa nella `strcat()`.

- *Hint 7: 8: [1] (buffer) strlen: Does not handle strings that are not \\0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).*

**Analisi:** falso positivo, non è possibile che avvenga un buffer overflow, il carattere di fine stringa viene considerato sommando il `+1`.

- *Hint 8: 9: [1] (buffer) strncpy: Easily used incorrectly; doesn't always \0-terminate or check for invalid pointers [MS-banned] (CWE-120).*

**Analisi:** non è possibile che avvenga un buffer overflow dato che la dimensione di `dst` viene calcolata in base a quella di `src`. Manca però il controllo per verificare se il puntatore `src` sia NULL oppure no, si risolve con un semplice controllo usando un `if`.

- *Hint 9: 9: [1] (buffer) strlen: Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).*

**Analisi:** falso positivo, non è possibile che avvenga un buffer overflow, poiché il carattere terminatore viene inserito alla fine della stringa nell'istruzione successiva, `dst[strlen(dst)] = 0`.

- *Hint 10: 10: [1] (buffer) strlen: Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) (CWE-126).*

**Analisi:** falso positivo, in quanto inseriamo il carattere terminatore proprio alla fine dell'array, nell'ultima posizione di `dst`.

- *Hint 11: 17: [1] (buffer) read: Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).*

**Analisi:** falso positivo, in questo caso la funzione `read()` non è usata all'interno di un loop, va comunque controllato se la funzione è andata a buon fine o meno.

- *Hint 12: 22: [1] (buffer) read: Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).*

**Analisi:** falso positivo, in questo caso la funzione `read()` non è usata all'interno di un loop, va comunque controllato se la funzione è andata a buon fine o meno.

- *Hint 13: 34: [1] (buffer) strncpy: Easily used incorrectly; doesn't always \0-terminate or check for invalid pointers [MS-banned] (CWE-120).*

**Analisi:** falso positivo, la stringa `buffer` ha sempre il carattere di fine stringa che `strcat()` cercherà e interpreterà.

#### 4 – Vulnerabilities, if any, not flagged by the tool

Analizzando il file ho individuato le seguenti vulnerabilità che non sono state rilevate dal tool Flawfinder:

1. Riga 41: la variabile `foo` potrebbe essere più grande di 10 e quindi va cambiata l'assegnazione di `buffer`, più precisamente va cambiata la `malloc`. Per risolvere questa vulnerabilità si agisce nel seguente modo:

```
char *buffer = (char *)malloc((strlen(foo)+1)*sizeof(char));
strncpy(buffer, foo, sizeof(buffer) - 1);
buffer[sizeof(buffer)-1] = '\0';
```

2. Riga 61: l'array `a` ha dimensione 10, quando il ciclo verrà eseguito per la prima volta si avrà `y=10` e quindi buffer overflow, va messo un controllo dopo il `while`. Per risolvere questa vulnerabilità si inserisce questo `if`, nel quale confronto `y` con il numero di elementi dell'array:

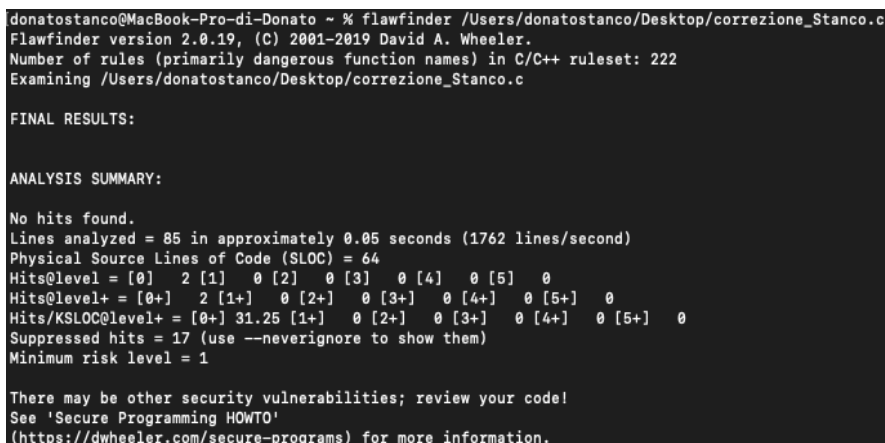
```
while (y>=0) {
    if(y < sizeof(a) / sizeof(int))
        a[y]=y;
    y=y-1;
}
```

3. Riga 30: funzione `fgets()`, va controllato il valore restituito dalla funzione nel caso in cui vi sia un errore.

```
if(fgets(buffer, 1024, stdin) == NULL){
    return;
}
```

**5 – A corrected version of the fragment where all the vulnerabilities found have been removed, with a screenshot showing that the tool “has no complains”.**

Facendo analizzare il codice corretto dal tool `FlawFinder` si avrà il seguente risultato:



```
donatostanco@MacBook-Pro-di-Donato ~ % flawfinder /Users/donatostanco/Desktop/correzione_Stanco.c
Flawfinder version 2.0.19, (C) 2001-2019 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 222
Examining /Users/donatostanco/Desktop/correzione_Stanco.c

FINAL RESULTS:

ANALYSIS SUMMARY:

No hits found.
Lines analyzed = 85 in approximately 0.05 seconds (1762 lines/second)
Physical Source Lines of Code (SLOC) = 64
Hits@level = [0] 2 [1] 0 [2] 0 [3] 0 [4] 0 [5] 0
Hits@level+ = [0+] 2 [1+] 0 [2+] 0 [3+] 0 [4+] 0 [5+] 0
Hits/KSLOC@level+ = [0+] 31.25 [1+] 0 [2+] 0 [3+] 0 [4+] 0 [5+] 0
Suppressed hits = 17 (use --neverignore to show them)
Minimum risk level = 1

There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(https://dwheeler.com/secure-programs) for more information.
```

Affinché il codice possa essere eseguito devono essere apportate le seguenti modifiche:

1. Per poter usare le librerie e quindi le varie funzioni in esse contenute è stato aggiunto il simbolo `#` prima della parola chiave `include`.
2. Per poter utilizzare le funzioni `read()` e `isalpha()` sono state aggiunte le librerie `unistd.h` e `ctype.h`.
3. La variabile `aFile` usata nel `main()` nella funzione `fprintf()` non era dichiarata, quindi è stata creata e successivamente gli è stato assegnato un file temporaneo che non crea problemi a livello di vulnerabilità, fornendo un hit di livello 1 che può essere ignorato, con l'istruzione `// flawfinder: ignore`.
4. Il costrutto `try/catch` presente nel `main()` è stato rimosso in quanto il linguaggio C non lo supporta. Per questo motivo la gestione dell'errore gestito usando `errmsg` presente nel `main()` è stato reindirizzato nella funzione `func3()`.

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

#include <unistd.h>
#include <ctype.h>

void func1(char *src)
{
    if(src != NULL){
        char dst[(strlen(src) + 1) * sizeof(char)]; // flawfinder: ignore
        strncpy(dst, src, strlen(src) + sizeof(char)); // flawfinder: ignore
        dst[strlen(dst)] = 0; // flawfinder: ignore
    }
}

void func2(int fd)
{
    char *buf;
    size_t len;
    if(read(fd, &len, sizeof(len)) == -1) // flawfinder: ignore
        return;
    if (len > 1024)
        return;
    buf = malloc(len+1);
    if(read(fd, buf, len) == -1) // flawfinder: ignore
        return;
    buf[len] = '\0';
}

void func3()
{
    char buffer[1024]; // flawfinder: ignore
    printf("Please enter your user id :");
    if(fgets(buffer, 1024, stdin) == NULL){
        return;
    }

    if (!isalpha(buffer[0]))
    {
        char errormsg[1044]; // flawfinder: ignore
        strncpy(errormsg, buffer, 1024); // flawfinder: ignore
        strcat(errormsg, " is not a valid ID"); // flawfinder: ignore
        fprintf(stderr, "'%s'", errormsg);
    }
}
```

```
void func4(char *foo)
{
    if(foo != NULL){
        char *buffer = (char *)malloc((strlen(foo)+1) * sizeof(char)); // flawfinder: ignore
        strncpy(buffer, foo, sizeof(buffer) - 1); // flawfinder: ignore
        buffer[sizeof(buffer)-1] = '\0';
    }
}

main()
{
    int y=10;
    int a[10];
    FILE *aFile;

    func4("fooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo");
    func3();

    aFile = tmpfile(); // flawfinder: ignore
    if(aFile == NULL){
        printf("cannot create this temporary file");
        return 0;
    }

    fprintf(aFile, "%s", "hello world");
    fclose(aFile);

    while (y>=0) {
        if(y < sizeof(a) / sizeof(int))
            a[y]=y;
        y=y-1;
    }
    return 0;
}
```