SAPIENZA
UNIVERSITÀ DI ROMA

Security in Software Applications Proj 1

Alex Parri - 2140961 - Master's Degree in Cybersecurity

A.Y. 2024/25

Abstract

This is the report for the **first project** of the Security in Software Applications course directed by Daniele Friolo for the Academic Year 24/25 for the Master's Degree in **Cybersecurity** at Sapienza University of Rome. In this homework, the goal was to use the flawfinder tool to **statically analyze** the provided code fragment, **reason** on all of the tool's reports, and finally output a **corrected** version where all found vulnerabilities are removed.

Flawfinder

The flawfinder tool is a **simple, fast and lightweight** command-line utility tool for scanning C/C+ code for known security vulnerabilities. It is great as a **first check** for quickly identifying possible security problems within the code before it is sent to **more thorough** analysis tools.

Its biggest **weaknesses** lie in the fact that it is limited to static analysis only, has a high **false positive** rate, has no context awareness, and that it cannot be relied upon on its own.

Making the code compilable

The provided C code is **not compilable**, due to multiple syntax errors, part of which are shown below

```
(proj1venv) alex@alex-MS-7C37:~/uni/ssa/proj1$ gcc project1_SSA24_compilable.cpp -o compiled_project1_SSA24
project1_SSA24_compilable.cpp: In function ‘void func2(int)’:
project1_SSA24_compilable.cpp:21:17: error: invalid conversion from ‘void*’ to ‘char*’ [-fpermissive]
   21 |     buf = malloc(len+1);
      |           ~~~~~~^~~~~~
      |                 |
      |                 void*
project1_SSA24_compilable.cpp: In function ‘int main()’:
project1_SSA24_compilable.cpp:47:11: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
   47 |     func4("fooooooooooooooooooooooooooooooooooooooooooooooooooo");
      |           ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
project1_SSA24_compilable.cpp:56:16: warning: format not a string literal and no format arguments [-Wformat-security]
   56 |         fprintf(stderr, message);
      |         ~~~~~~~^~~~~~~~~~~~~~~~~
(proj1venv) alex@alex-MS-7C37:~/uni/ssa/proj1$ gcc project1_SSA24.c -o compiled_project1_SSA24
project1_SSA24.c:2:9: error: expected ‘=’, ‘,’, ‘;’, ‘asm’ or ‘__attribute__’ before ‘<’ token
    2 | include <stdlib.h>
      |         ^
project1_SSA24.c: In function ‘func2’:
project1_SSA24.c:16:1: error: unknown type name ‘size_t’
   16 | size_t len;
      | ^~~~~~
```

In order to have a **more representative** output from the tool, it was decided to make the code compilable by fixing the syntax errors in the **most faithful way**.

Specifically, to preserve the `try...catch` construct, the code was converted into C++, as it does not exist in plain C. This is **not an issue** as the tool is able to analyze cpp files either way.

Utilizing the tool

The tool was then ran on the **compilable version** of the code to look for potential vulnerabilities

```
(proj1venv) alex@alex-MS-7C37:~/uni/ssa/proj1$ flawfinder project1_SSA24_compilable.cpp --neverignore
Flawfinder version 2.0.19, (C) 2001-2019 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 222
Examining project1_SSA24_compilable.cpp
Warning: Skipping non-existent file --neverignore

FINAL RESULTS:

project1_SSA24_compilable.cpp:40:  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination [MS-banned]
  (CWE-120). Consider using snprintf, strcpy_s, or strlcpy (warning: strncpy
  easily misused).
project1_SSA24_compilable.cpp: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.
project1_SSA24_compilable.cpp: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.
```

The following is a **rundown** of all the 13 detected warnings ordered by their **risk levels**

1. project1_SSA24_compilable.cpp:40 [4] (buffer) strcpy

Does not check for buffer overflows when copying to destination [MS-banned] (CWE-120). Consider using snprintf, strcpy_s, or strlcpy

2. project1_SSA24_compilable.cpp: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.

3. project1_SSA24_compilable.cpp:8 [2] (buffer) char

Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119!/CWE-120).

4. project1_SSA24_compilable.cpp:27 [2] (buffer) char

Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119!/CWE-120).

5. project1_SSA24_compilable.cpp:32 [2] (buffer) char

Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues (CWE-119!/CWE-120).

6. project1_SSA24_compilable.cpp:34 [2] (buffer) strcat

Does not check for buffer overflows when concatenating to destination [MS-banned] (CWE-120). Risk is low because the source is a constant string.

7. project1_SSA24_compilable.cpp: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).

8. project1_SSA24_compilable.cpp:9 [1] (buffer) strncpy

Easily used incorrectly; doesn't always \0-terminate or check for invalid pointers [MS-banned] (CWE-120).

9. project1_SSA24_compilable.cpp: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).

10. project1_SSA24_compilable.cpp: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).

11. project1_SSA24_compilable.cpp:16 [1] (buffer) read

Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).

12. project1_SSA24_compilable.cpp:22 [1] (buffer) read

Check buffer boundaries if used in a loop including recursive loops (CWE-120, CWE-20).

13. project1_SSA24_compilable.cpp:33 [1] (buffer) strncpy

Easily used incorrectly; doesn't always \0-terminate or check for invalid pointers [MS-banned] (CWE-120).

Undetected vulnerabilities

Corrected version

The following snippet contains the corrected version of the code with **comments** attached