Software Assurance Tips

A product of the Software Assurance Tips Team[2]

Chris Ramsey

Monday 7th July, 2025

1 The -Wall GCC Flag and CWE-457: Why -Wall Does Not Offer Sufficient Protection Against Uninitialized Variables

Updated Tuesday 27th January, 2026

1.1 Introduction

CWE-457 is a category of software weakness related to the use of uninitialized variables.[1] In some programming languages—C and C++ in particular—local (stack) variables are not automatically initialized. Instead, they contain garbage data, consisting of leftover values from previous operations. Using uninitialized variables can easily lead to unpredictable behavior and security vulnerabilities.

The `-Wall` command-line argument in the GNU Compiler Collection (GCC) enables a wide range of useful warnings, including some that help detect the use of uninitialized variables. For example, consider the program in Listing 1. When compiled with the GCC arguments `-Wall` and `-Werror`, the uninitialized value is properly detected as shown in Listing 2.

```
#include <cstdio>

int main() {
        int x;
        int y = x + 1; // 'x' is used uninitialized
        printf("%d\n", y);
        return 0;
}
```

Listing 1: Simple Uninitialized Variable

```
main.cpp: In function ‘int main’():
main.cpp:5:9: error: ‘’x is used uninitialized [□Werror=uninitialized]
    5 |     int y = x + 1; // 'x' is used uninitialized
      |         ^
main.cpp:4:9: note: ‘’x was declared here
    4 |     int x;
      |         ^
cc1plus: all warnings being treated as errors
```

Listing 2: Compilation with -Wall and -Werror

This has led to a common misconception: that the `-Wall` flag offers developers sufficient protection against CWE-457. However, this is not the case. In this SwA tip, we explore some common situations where `-Wall` fails to detect the use of uninitialized variables.

Example 1: Conditional Assignment

Consider the example in Listing 3. Compiling this program with GCC version (Ubuntu 13.3.0-6ubuntu2 24.04) 13.3.0 using the `-Wall` command-line argument produces no warnings. However, running the program a few times is usually sufficient to demonstrate the issue. Sometimes, it does not output `10` (see Listing 4).

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

int main() {
        int x;
        srand(time(NULL));
        if (rand() % 2 == 0) { // 50% chance of being true
                x = 10;
        }
        printf("%d\n", x);
        return 0;
}
```
Listing 3: Conditionally Uninitialized Variable

```
./main.exe
10
./main.exe
10
./main.exe
32765
```
Listing 4: Execution Showing Uninitialized Variable

Example 2: Function Scope Escape

Consider the example in Listing 5. Using the same compiler as Example 1, GCC does not detect that variable `x` remains uninitialized after the call to `foo`. Passing a pointer to the variable is enough to fool the compiler.

```
#include <cstdio>

void foo(int* x) {
        // does nothing
}

int main() {
        int x;
        foo(&x);
        printf("%d\n", x);
        return 0;
}
```
Listing 5: Function Scope Escape Uninitialized Variable

Example 3: Uninitialized Class Member

Consider the example in Listing 6. The private member variable `m_x` is used uninitialized without any warning from the same GCC version as the previous examples.

```
#include <cstdio>

class MyClass {
        int m_x;

        public:
                void print() {
                        printf("%d\n", m_x);
                }
};

int main() {
        MyClass obj;
        obj.print();
        return 0;
}
```

Listing 6: Class Member Uninitialized

Conclusion

While the `-Wall` flag does provide some protection by detecting simple instances of CWE-457, many common cases remain undetected. The SwA team generally recommends developers follow industry best practices—such as always initializing variables—and use static analysis tools whenever practical to help detect this issue.

References

[1] CWE Content Team. “CWE-457: Use of Uninitialized Variable”. In: (2025). URL: `https://cwe.mitre.org/data/definitions/457.html`.

[2] Jon Hood, ed. SwATips. `https://www.SwATips.com/`.