Sure! Here’s an example of C# code that demonstrates an integer overflow vulnerability, along with detailed explanations:

**C#**

using System;

class Program

{

static void Main()

{

int maxValue = int.MaxValue; // Maximum value an int can hold: 2,147,483,647

int result = maxValue + 1; // This causes an overflow

Console.WriteLine("Max Value: " + maxValue);

Console.WriteLine("Result after overflow: " + result);

}

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new" \l "faq).

### Explanation:

1. Initialization:

**C#**

int maxValue = int.MaxValue;

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new" \l "faq).

* + **int.MaxValue** is a constant in C# that represents the maximum value an **int** can hold, which is 2,147,483,647.

1. Overflow Occurrence:

**C#**

int result = maxValue + 1;

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new" \l "faq).

* + Adding 1 to **maxValue** causes an overflow because the result exceeds the maximum limit of an **int**. In C#, this wraps around to the minimum value an **int** can hold, which is -2,147,483,648.

1. Output:

**C#**

Console.WriteLine("Max Value: " + maxValue);

Console.WriteLine("Result after overflow: " + result);

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new" \l "faq).

* + The program prints the maximum value of an **int** and the result after the overflow. The output will be:

Max Value: 2147483647

Result after overflow: -2147483648

### Why is this a vulnerability?

An integer overflow can lead to unexpected behavior and potential security risks. For example, if this overflow occurs in a critical part of the application, it could be exploited by an attacker to cause a denial of service or to manipulate the program’s behavior in unintended ways.

### How to prevent it?

1. Checked Context: Use the **checked** keyword to explicitly enable overflow checking, which will throw an exception if an overflow occurs.

**C#**

int result;

try

{

result = checked(maxValue + 1);

}

catch (OverflowException)

{

Console.WriteLine("Overflow detected!");

}