The provided C++ code is vulnerable to the following security issues:

**1. Buffer Overflow:**

* The arr array is declared with a fixed size n. If the user inputs a value for n that is larger than the allocated memory, it could lead to a buffer overflow, potentially allowing an attacker to execute arbitrary code.

**2. Integer Overflow:**

* If the user inputs a very large value for n, it could cause an integer overflow when calculating array indices or memory allocations. This could lead to unexpected behavior or security vulnerabilities.

**3. Input Validation:**

* The code does not perform any input validation on the user-provided n and the elements of the arr array. Malicious input could lead to various security issues, such as buffer overflows, format string vulnerabilities, or injection attacks.

**4. Memory Leaks:**

* While the code doesn't explicitly allocate memory dynamically, it's important to note that if dynamic memory allocation were used, improper memory management could lead to memory leaks, which can degrade system performance and potentially open security vulnerabilities.

To mitigate these vulnerabilities, consider the following:

* **Input Validation:** Implement robust input validation to ensure that user-provided input is within expected bounds and doesn't contain malicious content.
* **Safe Memory Allocation:** If dynamic memory allocation is necessary, use techniques like malloc, calloc, and realloc carefully, and always free allocated memory when it's no longer needed.
* **Secure Coding Practices:** Adhere to secure coding principles to avoid common vulnerabilities like buffer overflows, integer overflows, and format string attacks.
* **Code Review:** Conduct thorough code reviews to identify and address potential security issues.
* **Security Testing:** Perform regular security testing, such as penetration testing, to uncover vulnerabilities and weaknesses in the code.

By addressing these vulnerabilities, you can significantly enhance the security of your C++ program.