The C++ programs shared can be vulnerable to several security issues if not carefully managed:

1. **Input Validation**: The code doesn't validate user input effectively. For instance, when taking the size of the array, if the user enters a negative number or a very large number, it could cause undefined behavior or memory allocation issues.
2. **Integer Overflow**: If the user enters a large enough integer for the array size, it might cause an integer overflow, leading to incorrect memory allocation.
3. **Memory Leaks**: While the programs do deallocate memory, it's crucial to ensure that every path in a more complex program deallocates memory properly. Forgetting to deallocate memory after certain operations could lead to memory leaks.
4. **Buffer Overflows**: Without proper bounds checking, there's a risk of buffer overflows. If the user input exceeds the allocated memory space, it might corrupt adjacent memory, leading to undefined behavior or security vulnerabilities.
5. **Dangling Pointers**: If memory is deallocated but the pointer is still in use somewhere else in the program, this could lead to dangling pointer dereference, causing crashes or unexpected behavior.
6. **Denial of Service**: The user can allocate a very large amount of memory, causing the system to run out of memory and potentially leading to a denial of service.

In more comprehensive and secure applications, additional measures like input validation, exception handling, and robust memory management practices are essential to mitigate these vulnerabilities.