A reflection on a case of an air traffic control systems failure

In 2014, a glitch in an air traffic control system caused hundreds of delayed flights in the Los Angeles area. Read about the incident here: https://www.reuters.com/article/us-airtraffic-bug-exclusive/exclusive-air-traffic-system-failure-caused-by-computer-memory-shortage-idUSBREA4B02320140512

A lack of altitude information for a plane meant that the system could not calculate its flight path. Without this information, the system got stuck in a loop running calculations for all possible altitudes from 0 – infinity. This loop caused the system to run out of memory, triggering a system restart. The same issue would occur again however when the system turned back on leading to a cycle or restarts.

Thoughts on how this could have been avoided

1. The use of better test scenarios – The testers should have tested for invalid and missing data

2. Treat the flight's plans as untrustworthy – The flight plan info should have been considered unsafe there for, malicious input, such as missing data should have been handled.

3. Defensive programming should have been used- the systems code should have used defensive programming to check all required data was provided.

4. Preallocate memory for each flight – If each flight had a pre-allocated memory size, then the memory usage would be proportional to the number of flights. As such it would be much easier to predict an error before it occurred. In this scenario, the worst case a specific limited number of planes could be delayed to a time when the system has more free memory.

5. Control structures without a base condition should not be used. If a base condition had been used to trigger an exception then this error would not have occurred.

6. Prefer iterative control structures to recursive because iterative control structures have a constant memory size.

Issues caused by this failure

1. This failure has exposed the flight plan data as an attack surface which may prompt more attacks.

2. People may distrust flying

3. People may distrust the software profession

4. People's flights where missed and this could cause all sorts of problems for those people

Finally, I don't think that the solution used by the system provider of adding more memory was a good choice, as for example an infinite for loop will use an infinite memory. Instead, they team should look to identify all attack surfaces and test any potentially malicious inputs. Further the team should look to better control memory usage so it is predictable or constant.