Paper summary

Instead of throwing an exception or terminating in failure oblivious computing, we use the safe-C compiler to ignore it and continue the execution. The safe-C compiler inserts check for invalid memory references or reads an out-of-bound array. The idea may sound bizarre but has its merit-Programs can run unless address space or data structure gets corrupted. It helps to enhance the security and availability of the system. Debugging is always the first preference, but it may not be possible because of a lack of time or source code. In a buffer overrun attack, the server discards the erroneous write and generates value for erroneous reads. It doesn't terminate in case any exception occurs and continues to run. It can use it in an environment where the server cannot afford to crash and in an interactive environment such as mailers and document processing systems. A memory failure can address exceptions, infinite loop, change in control flow, corrupt data structure, or produce unacceptable results. Failure-oblivious helps address all these issues and converts dangerous attacks to known cases.

Strengths

1. It sounds like an exciting idea in scenarios where the programmer does not handle exceptions and bound checks. If a bug occurs in production, the safe-C compiler can prevent it.

2. The scope and drawbacks are well defined.

3. The evaluation criteria were well executed and thorough. If it is deployed and still working, there should be merit to it.

4. Increase availability and better security.
5. Less human efforts.

Weaknesses

1. Unexpected behaviour of the program.

2. It cannot be used in safety-critical applications.

3. Developers can become sloppy.

Comments for author

Even though I strongly agree that the onus of responsibility should be on the programmer to handle exceptions and illegal memory addresses, humans can be sloppy. This system can be used where halting is not affordable, but other than that, I would still prefer to use a crash-only mechanism that can handle failure by simply restarting. Additionally, I think there could be more discussion of the variants and extensions and just how this system could be used for other applications than the ones mentioned.