Document #: P2272R0
Date: 2020-12-08
Programming Language: Modern C++
Audience: WG21 Directions Group, EWG, LEWG, SG12, SG14
Reply to: Matthew Butler (mbutler@laurellye.com)

Safety & Security Review Group

Matthew Butler

1 Forward

Two areas where Modern C++ struggles to maintain its dominance is in the areas of safety critical and secure applications. The complexity of the language and the perception that C++ is unsafe and insecure gives other languages, such as Rust and Golang, an edge. While the move to zero overhead abstractions and more modern development techniques has helped, there is still significant work to be done addressing C++'s perceived and real shortcomings in the areas of safety and security.

Moreover, these areas often involve embedded systems where C++ is also struggling with both a lack of updated tooling and performance perceptions. Embedded systems also carry their own special requirements which are generally covered by SG14 but should be considered as part of the safety and security focus of this group.

2 Safety Critical and Secure Applications

So what do we mean by Safety Critical and Secure applications?

Safety Critical applications are applications where human safety is at risk. This includes airplanes, autonomous vehicles, medical devices, et al. The primary objective of a safety critical application is the preservation of human life, the secondary objective is the original intent of the application: moving people from point A to point B, delivering life saving medical treatment, etc.

Safety critical applications usually, but not always, involve embedded systems with strict timing requirements. One example is the airbags in a vehicle that must deploy under what are known as hard real-time requirements. Deploy too late and they fail to protect the occupants, deploy too early and they begin deflating early, losing their effectiveness.

The ultimate requirement for safety critical applications is the preservation of human life.

For more information see *Modern C++ Safety & Security At 20* *(CppCon 2020).*

Secure applications, on the other hand, have the primary objective of protecting data and capabilities. This may be the Personally Identifying Information (PII) for millions of people, the plans to the Death Star or the national power grid. A secure application is one that has few, if any, vulnerabilities that can be exploited. Unlike safety critical applications, secure applications cross all platforms: embedded, general computing, HPC, GPU, et al.

When applications are “insecure”, they contain vulnerabilities that can be exploited by threat actors to gain access to the protected data and exfiltrate it or make the system act in a way that is contrary to its intent. Destroying the power grid by forcing equipment to behave destructively, for example.

The ultimate requirement for Secure applications is the preservation of data and capabilities.

For more information see *Secure Coding Best Practices* *(CppCon 2018).*

So how are they related?

By definition, all safety critical applications have very high security requirements. Imagine, for example, someone taking over a jet remotely by exploiting a software vulnerability. And, for the most part, safety critical and Secure applications solve the same problems. They are both interested in:

- Memory safety
- Exception handling
- Determinism
- Undefined Behavior
- Algorithm correctness
- Provability
- Code quality
- Real-time requirements (generally safety critical only)

While all applications share some or all of these requirements, the costs of failure are highest when the consequences are highest: loss of human life, loss of sensitive data, destruction of the Death Star. Safety critical and secure applications push each of these areas to the absolute limit because the costs of failure are highest.

3 Proposal

As discussed, some areas where C++ has a role to play in the areas of safety and security are:

- Aerospace
- Autonomous vehicles
- Networking and networking security devices
- Medical devices
- Communications
- Industrial control systems
- Cloud

Note that all of the above areas have both safety and security requirements, involve both embedded and general-purpose systems and deal with high performance systems. All also suffer from the perception that C++ is unsafe, and all areas have languages which are considered “safer” alternatives that are challenging C++ for dominance in their respective markets.

To combat those perceptions, the C++ Standards Committee should make a concerted effort to address these issues in the language standard itself.

To that end, it is proposed that the committee create a safety and security review group (similar to the ABI Review Group). As safety and security are inextricably linked and often deal with many of the same issues, the board would cover both areas with a subspecialty in embedded systems as they apply to safety and security.

The board’s mandate would be to:

- *evaluate new language and library features for safety & security, received from the various SGs including EWG and LEWG (There are already a number of proposals in flight from these groups and they require a specialized group with the expertise to evaluate them),*

- *advise the Directions Group and the individual Study Groups on matters of safety & security in future directions, and*

- *recommend changes to the existing specification to address the perceived and real issues of safety & security.*

The *Safety & Security Review Group (SSRG)* would be an invitation-only group of industry experts from across the safety critical, systems security and embedded communities. The SSRG’s mandate would be to act in an advisory capacity for WG21 and would not initiate changes to the language standard itself. That role belongs to the SGs themselves.

In this way the SSRG is a passive organization. Where the SSRG can be an active organization is in the areas of outreach to the ISO committee itself and education for both the community and ISO committee to help them understand the implications of language design decisions and language choices. It would also work with other ISO groups where changes also potentially originate, namely WG23 and WG14.

4 Organization

The board would be organized with a single chair, appointed by the convening authority, who ideally has experience in the safety critical, security and embedded areas, who would schedule the monthly meetings and organize the board. The board members should be able to cover safety critical applications, security, and embedded areas.

Having current chairs from existing Study Groups on the board is advantageous because it allows cross pollination into their respective SGs. The size of the board should be limited to something on the order of 5-10 members (no more than one representative per company) plus the chair to make scheduling possible given the number of ISO meetings we currently hold.

Note that the SSRG is not intended as a replacement for or as a competitor with SG12 or SG14. Indeed, the work of the SSRG cuts across all SGs. It will likely have members from the respective SGs for the purpose of supporting the paper authors themselves along with the committee.

5 Areas of Focus

The primary areas that the group would focus its attention on would be aligned with the subject matter expertise of the group's members, namely:

- *exploitability of new and existing core language and library features*
- *suitability of new core language and library features for safety critical and secure applications*
- *definition of "approved" language subsets for safety critical and secure applications*
- *strategic recommendations for the DG covering core language and library updates in the areas of safety critical and secure applications*
- *educating the various SGs on the impact of language definition on the areas of safety critical and secure applications*
- *educating the community on safe coding standards for both safety critical and secure applications*

The process by which all of this will be accomplished will be determined once the group has been formed and a chair appointed.

6 Final Thoughts

The ultimate goal for this group is to help guide the development of Modern C++ towards a safer language that still has the speed and power we need for high performance applications.

Where the language can be changed to support embedded, safety critical and secure applications, this group can help define those areas. Where it cannot, this group can help define a subset of the language standard that directly supports safety critical and secure applications development.

7 References

"C++ Can Affect Lives" Bjarne Stroustrup, 2019 Cpp Summit keynote

"Modern C++ Safety & Security at 20" Matthew Butler, 2020 CppCon

"Secure Coding Best Practices" Matthew Butler, 2018 CppCon

"What Air Disasters Tell Us About Safety Critical Designs" Matthew Butler, 2020 Cpp Summit

"Exploiting Modern C++: Writing Secure Code for an Insecure World" Matthew Butler, 2021 Pearson Publishing