BLACK DUCK®

GUIDE

ENSURING QUALITY IN MISSION-CRITICAL EMBEDDED SYSTEMS

Embedded software is increasingly mission-critical in a variety of industries. Autonomous vehicles use it to assess traffic patterns, respond to road conditions, and protect passengers. Medical devices such as infusion pumps and ECG machines rely on it to monitor vital signs and control key functions. It is clearly imperative that the embedded software made for these types of use cases is highly reliable and predictable throughout the full life cycle of the devices that contain it.

This guide outlines the challenges of producing high-quality, robust embedded software for mission-critical systems. It also explores solutions that can help organizations ensure that their software is free of critical defects and security vulnerabilities.

CHALLENGES TO PRODUCING HIGH-QUALITY EMBEDDED SOFTWARE

The Complexity of Software

The sheer complexity of large-scale software projects is the #1 challenge developers face when producing embedded software. This complexity can make it difficult to identify some types of defects and vulnerabilities, particularly those that span multiple files or libraries. Additionally, many large-scale software projects use programming languages and frameworks that introduce their own challenges.

For example, the C++ programming language provides advantages due to its versatility and performance. However, the complexity and low-level nature of the language can introduce specific challenges, such as

- **Memory management errors:** Manual allocation and deallocation (e.g., new/delete) can lead to memory leaks, dangling pointers, and double frees
- **Buffer overflows:** The lack of bounds-checking in arrays and pointer arithmetic can allow attackers to overwrite memory
- **Unexpected behavior:** C++ allows operations (e.g., using uninitialized variables) that can result in unpredictable behavior if not carefully managed

Embedded software also faces challenges such as hardware failure and the frequent need to interact with external systems or the broader environment. It must operate in environments where data is passed from one system to another, sometimes across on-premises and cloud deployments. In this scenario, any component could fail, malfunction, or introduce vulnerabilities that could become vectors for attacks. And all these challenges can be exacerbated when the software includes open source components, which can introduce defects and vulnerabilities if not properly managed.

BALANCING QUALITY AGAINST SPEED

Development teams are under constant pressure to deliver new features to satisfy customers, keep up with competition, and increase market share. Teams must also divide their time between writing code and requests from other business units, training on tooling or best practices, and lengthy triage sessions to address issues identified in quality or security tests. And when critical defects or vulnerabilities are detected late in the development life cycle, teams may be forced to compromise between hitting a release date or shipping software that doesn't meet quality or security standards.

New Technology Advancements

Widespread adoption of AI has very quickly translated to improved development velocity, particularly due to AI coding assistants. But although these tools reduce the burden on developers to write code from scratch, they also increase the likelihood of errors or security issues, because AI coding assistants are trained on open source and sample code that can have significant vulnerabilities in it. And to further complicate the issue, in some cases, the speed of development has increased so much that it's outpacing testing capacity, increasing the risk of low-quality or insecure code making it into production.

AI coding assistants also increase risk due to the lack of transparency into the origin and quality of training data, their decision-making processes, and security controls. Nevertheless, as the use of these tools increases, large portions of software projects will be written by AI. So it's imperative that organizations put measures in place to identify and mitigate critical defects and vulnerabilities in AI-generated code.

Regulatory Requirements

There are many industry-specific laws and regulations that govern the quality and security of software. These regulations are aimed at ensuring consumer safety, cybersecurity, and reliability, and seeing that appropriate protocols are adopted to help protect critical networks and infrastructure.

In the automotive industry, for example, ISO/SAE 21434 specifies development requirements and processes to reduce risks and ensure driver safety throughout the life cycle of road vehicles. In the healthcare industry, there are several regulations focused on ensuring the safety and efficacy of medical devices and protecting patient data. And as 5G technologies drive new advancements, the telecom industry is adopting Open Radio Access Network standards that promote the use of interoperable and open source components in networks.

BUILDING ROBUST SOFTWARE THAT'S FREE OF CRITICAL DEFECTS

Comprehensive Static Analysis at Scale

In-depth code analysis is used to identify critical defects or vulnerabilities that may not be detected by other scans. In very large projects, issues can be particularly difficult to detect, especially when they span multiple files. But in-depth testing can yield tens of thousands of findings. If these findings are littered with false positives, developers end up wasting valuable time triaging issues, and prioritization becomes very difficult, if not impossible.

Coverity® Static Analysis can uncover complex issues in even the largest codebases. It provides insights into all dependencies and compilers, and it offers support for more than 20 programming languages and 200 frameworks. Coverity builds a detailed model of each application to analyze dataflow, control flow, and semantics. Its validation component collects evidence about potential defects and the context in which the code is used, minimizing false positives. This yields highly accurate results that can be mapped to a broad range of regulatory standards and prioritized for remediation.

Protocol Fuzzing

While static analysis relies on access to the codebase to identify specific types of issues based on predefined rules, protocol fuzzing uses an automated testing technique that detects crashes or unexpected behavior to identify potential defects or vulnerabilities. Malicious hackers and security researchers alike use fuzzing to trigger application failures and discover new vulnerabilities.

Defensics® Fuzzing uses an intelligent and targeted approach that sends as many invalid inputs as possible to a system to trigger defects such as resource leakage, busy loops, and crashes. Identifying these issues before software is released can prevent dangerous malfunctions and zero-day attacks.

The Defensics library consists of more than 300 prebuilt fuzz testing suites, which are maintained by a team of Black Duck engineers to help organizations get up and running quickly. Defensics is compatible with a broad range of communication protocols—a key integration point of connected systems.

Automated Scanning

Identifying quality and security issues early in the development life cycle is essential. When critical defects and vulnerabilities go undiscovered until shortly before software is released, development teams may have to make difficult compromises between quality and timeliness. Automated scanning is a tried-and-true approach to identifying and mitigating issues early in the software development life cycle. Coverity delivers

- **Real-time IDE scans:** Identify defects and vulnerabilities in real time as the developer writes code, and before it's committed
- **Automated scans on SCM events:** Trigger code scans to run on every pull request to identify issues in any new or changed code
- **Periodic full project scans:** Set up regular, in-depth scans to identify all issues in a project, while also checking code for compliance
- **Quality gate integration:** Ensure that no critical issues make it into production by blocking promotion or breaking builds if critical issues or policy violations are found

ENSURING STANDARDS COMPLIANCE

Organizations in highly regulated industries are often required to perform certain types of software tests on a regular basis. Static code analysis and fuzzing are among the types of testing that are often required to ensure the reliability of this software.

Coverity automatically maps defects and vulnerabilities to the specific standards that they violate. This simplifies triage and prioritization efforts, so development teams can resolve the most important issues as quickly as possible.

- **Identify relevant standards:** Determine which coding standards, such as MISRA, CERT C/C++, and OWASP, are most important to your organization and industry
- **Define policies:** Set policies to automate key compliance processes; for example, defining a policy that all code must be free of CERT C/C++ security vulnerabilities
- **Automate issue-mapping:** Integrate test results into your development workflows, enabling defects and vulnerabilities to be mapped to relevant standards to simplify triage and prioritization efforts

CONCLUSION

Embedded software has become a critical part of the products people interact with every day, so the quality of the code is more important than ever. As AI usage expands, development is getting faster while release cycles are getting shorter. Automating code quality and security testing as well as regulatory compliance management will be critical to keeping pace.

Coverity and Defensics help organizations identify even hard-to-find issues early in the development process to deliver compliant and robust embedded software at the speed today's businesses require.

ABOUT BLACK DUCK

Black Duck® meets the board-level risks of modern software with True Scale Application Security, ensuring uncompromised trust in software for the regulated, AI-powered world. Only Black Duck solutions free organizations from tradeoffs between speed, accuracy, and compliance at scale while eliminating security, regulatory, and licensing risks. Whether in the cloud or on premises, Black Duck is the only choice for securing mission-critical software everywhere code happens. With Black Duck, security leaders can make smarter decisions and unleash business innovation with confidence. Learn more at www.blackduck.com.