GRAMMATECH

DESIGNING SECURITY INTO MEDICAL DEVICE SOFTWARE

TRUSTED LEADERS OF SOFTWARE ASSURANCE AND ADVANCED CYBER-SECURITY SOLUTIONS

WWW.GRAMMATECH.COM

BACKGROUND

Considerable effort in the development of medical devices is focused on safety and reducing patient risk. Even so, with the recent increased connectivity of devices, security researchers have found security lacking in medical devices, with one recent example finding over 1400 security vulnerabilities in a commonly-used infusion pump.

As a reaction to this troubling trend, the Federal Drug Administration (FDA) has issued guidance on managing security in medical devices. This paper discusses how to refocus medical device development to include security as a key factor, in addition to safety.

FDA GUIDANCE ON MANAGING CYBERSECURITY AND STATIC ANALYSIS

The FDA, recognizing the need for more robust security in medical devices, issued its guidance on managing cybersecurity in 2014. The growth of wireless, networked, and Internet-connected devices means that medical devices are more at risk than ever before. In addition, medical devices deal with patient safety and privacy unlike other classes of devices. Risk management (including security hardening and vulnerability management) is the cornerstone of medical device software development — and static analysis plays a key role in the process.

Home health care and medical "wearables" are growing exponentially and are just one area of growth for medical devices. As with other medical and IoT opportunities, there are safety, security and privacy concerns associated with this growth.

The FDA guidance is quite broad and is meant to be a high-level guide for managing security. It includes many reasons for adopting automated tools, including the following:

- "Manufacturers should address cybersecurity during the design and development of the medical device."

  This is something GrammaTech has been communicating for some time — building in security from the start, rather than as an add-on, is key. More on this below.

- "The design and development approach should appropriately address identification of assets, threats, and vulnerabilities."

  Static analysis integrates seamlessly with good software development processes and specifically aids in detection and identification of security vulnerabilities in code and binaries.

- "Assessment of the impact of threats and vulnerabilities on device functionality and end users/patients and the likelihood of a threat and of a vulnerability being exploited."

  Using tainted data analysis, GrammaTech CodeSonar can trace the source of data throughout the software to indicate potential vulnerabilities from outside sources.

- “In the premarket submission, manufacturers should provide documentation related to the cybersecurity of their medical device.”

  Static analysis tools provide reporting tools to assist in process documentation and test completion and software readiness.

SECURITY-FIRST DESIGN

Security has not always been a primary concern for medical devices — connectivity among devices was for a long time assumed to be local, and in the hands of trusted operators and devices. Modern devices, however, are connected to networks (and frequently the Internet), so these devices require more serious attention to security and privacy. It is necessary to apply security principles earlier in the development lifecycle.

SOFTWARE SECURITY IN THE SOFTWARE DEVELOPMENT LIFECYCLE

A security-first design approach means integrating security as a top priority in the software development lifecycle (SDLC) as shown in the following diagram:

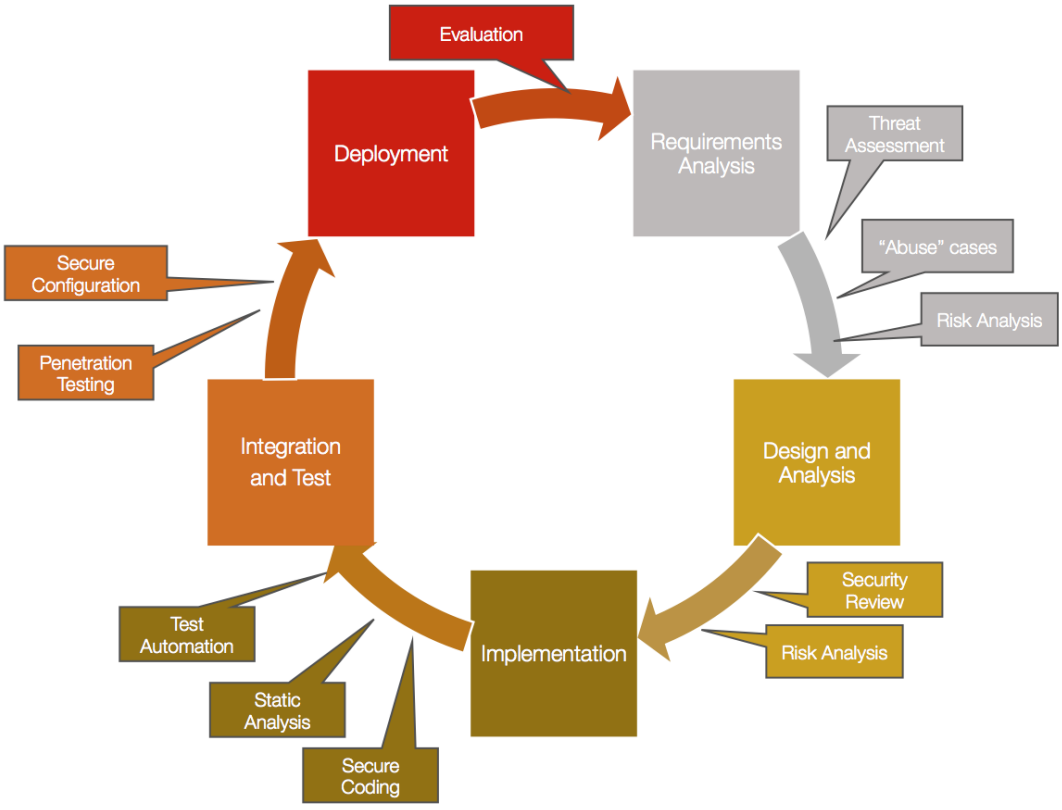

Figure 1: Security processes superimposed over the software design lifecycle.

Developers and project managers can expect at least the following types of activities at these key stages:

REQUIREMENTS STAGE

Once a system-wide threat assessment is available, the device threat surface can be understood. At the requirements stage, security-specific requirements can be introduced, along with known “abuse cases” (use cases that an attacker might follow) and a risk analysis. Security requirements, as listed below, are introduced and accounted for. This stage is critical because it is the point at which security becomes a known development project goal with the appropriate level of risk management, scheduling, and costing.

DESIGN AND ARCHITECTURE

As candidate architectures become available, reviews must include security aspects (where previously they may not have). Assessing architecture in light of the known threat assessment and security requirements adds an additional dimension to this phase of development. At this stage, testing plans should be created that include security analyses that follow the perceived “abuse cases.”

CODE DEVELOPMENT

At the coding stage, following security guidelines and coding standards are critical. The use of automation tools such as static analysis is key to ensure that vulnerabilities are not introduced into the product. Testing and test automation (that includes a security analysis) are important at this stage.

INTEGRATION AND TEST

As the system as a whole starts to take form, subsystem and system testing will find vulnerabilities before integration and deployment to the market. Automated penetration testing tools can be very helpful at this stage, to uncover vulnerabilities that may not have been accounted for in earlier stages of development. Packaging and configuration of the end product for deployment is key at the final stages of this phase. Ensuring that the out-of-the-box product is as secure as possible prevents many of the security issues we see today in connected devices.

DEPLOYMENT AND MAINTENANCE

When a product enters the market and starts wide deployment, security vulnerabilities become exponentially more costly to fix. A product designed with a security-first approach is less likely to end up with a security breach, but companies must be prepared to deal with security on an ongoing basis. Designing the product with the ability to update firmware and software is critical to addressing new-found issues expeditiously. However, as a product goes through maintenance and revision, security is an on going concern and new vulnerabilities and new threats need to fed back in to the system in an iterative approach.

SECURITY REQUIREMENTS

Securing a medical device requires many considerations. Key examples of security requirements that might go above and beyond existing functional requirements are as follows:

- User Authentication: validating user access and enforced privileges for different classes of users.
- Tamper Resistance: preventing physical and software changes to the device that allow circumvention of security functions.
- Secure Storage: ensuring stored data is protected from online and offline access, including techniques such as encrypted file storage and Digital Rights Management (DRM).
- Secure Communications: keeping data-transfer secure but also preventing unwanted access through connected channels (network, USB, etc.). Although network connectivity is top of mind, other channels are vulnerable to attack..
- Reliability and availability: maintaining safe operation of the device in the face of ongoing attacks.

APPLICABILITY OF SAST TOOLS TO IEC 62304 AND MEDICAL DEVICE SOFTWARE

Although the IEC 62304 standard doesn't call out specific development tools, it does indicate the need for rigorous testing, acceptance criteria, and traceability. Performing these functions without tools isn't practical given the scope of most medical device software projects. For example:

> **SECTION 5.5.2** requires a software unit verification process:
>
> *The MANUFACTURER shall establish strategies, methods and procedures for verifying each SOFTWARE UNIT.*
>
> **SECTION 5.5.3** requires:
>
> *The MANUFACTURER shall establish acceptance criteria for SOFTWARE UNITS prior to integration into larger SOFTWARE ITEMS as appropriate, and ensure that SOFTWARE UNITS meet acceptance criteria...does the software code conform to programming procedures or coding standards?*
>
> **SECTION 5.5.4** provides additional acceptance criteria:
>
> *When present in the design, the MANUFACTURER shall include additional acceptance criteria as appropriate for: proper event sequence; data and control flow; planned resource allocation; fault handling (error definition, isolation, and recovery); initialization of variables; self-diagnostics; memory management and memory overflows; and boundary conditions.*

Many of these acceptance criteria are well suited to Static Application Security Testing (SAST) a.k.a. static analysis tools. In fact, the case for static analysis is so strong, the FDA has used

GrammaTech CodeSonar to analyze medical device software to evaluate the quality of the source code following a series of infusion pump failures.

THE ROLE OF SAST TOOLS IN DESIGNING FOR SECURITY

SAST tools provide critical support in the coding and integration phases of development. Ensuring continuous code quality, both in the development and maintenance phases, greatly reduces the costs and risks of security and quality issues in software. In particular, it provides some of the following benefits:

- Static analysis finds bugs that coverage-based testing techniques miss.

  Unit testing is often measured by the level of coverage, such as statement or decision coverage. Although rigorous, there are defects that make it through this type of testing that static analysis tools can discover.

- Static analysis detects defects early.

  Preventing defects from occurring at the developer's desktop is the ideal situation — it saves money in testing and remediation that increases as a project progresses.

- Static analysis can handle SOUP.

  Software of Unknown Pedigree/Provenance (SOUP) requires special handling in medical device software, and good static analysis tools are capable of evaluating the quality and security of third-party and commercial off-the-shelf software (including binary-only executables and libraries).

- Static analysis accelerates pre-market approval documentation.

  Documenting the results of software unit acceptance is critical to proving compliance to certification standards. Static analysis tools have rich reporting features to help support FDA requirements.

CONCLUSION

Security is now a top-level risk/liability management factor in medical device software development. Designing security into the product at the early stages is both mandated by the FDA and good practice. SAST tools play an important part in accelerating the time-to-market for medical devices and assist in pre-market approval processes.

GrammaTech, Inc. is a leading developer of software-assurance tools and advanced cyber-security solutions. GrammaTech helps organizations develop and release high quality software, free of harmful defects that cause system failures, enable data breaches, and increase corporate liabilities in today's connected world. GrammaTech's CodeSonar is used by embedded developers worldwide.