Threat Modeling in Medical Software Development

A Thorough Review of Challenges and Benefits

ZEISS

Seeing beyond

zeiss.com/digital-innovation

How can Continuous Threat Modeling help reduce risks and increase efficiency in developing medical software? What are the challenges associated with a regulated environment? Discover in this whitepaper how Threat Modeling works as a proactive approach to ensure Security of Medical Software development – and its benefits. This publication provides a general overview, identifies challenges and an example of software development in a regulated environment.

Security in the medical context

As medical solutions and devices directly impact people's health the highest medical and security standards apply. Due to their nature, these products and solutions interact with and generate personal data, not only PII (Personal Identifiable Information), but also PHI (Protected/Personal Health information), which could be information about a user's medical conditions or surgery recordings. Of course, this is sensitive data, that comes with an extra need for security measures on the technical and regulatory side.

General security standards like ISO 27001, the NIST SP800-218 framework and the GDPR, as well as specific medical software standards, for example, HIPAA, or the QSR require higher level requirements like systematic risk analysis and security as a part of the design process. According to the newly released US-FDA-Guidance for Cybersecurity in Medical Devices for example, risk analysis and fitting measures as well as procedures to assure and maintain a cybersecure product are required.

Besides high-level requirements, standards and regulations also impact software requirements. However, these requirements are usually not broken down into case specific security requirements for software. For example, while GDPR obliges us to maintain adequate technical and organizational measures appropriate to the inherent risk, including pseudonymization and encryption of personal data, it is up to us to make sure that our development workflows include such risk assessments that allow us to derive appropriate protection measures.

In recent years, several general design principles have emerged which describe properties and paradigms to follow when designing and developing secure software. The following list summarize some of the most important concepts.

- Security and Privacy by Design
  Consider Security Threats and Data Protection Requirements in the design phase of Product Development and make it an inherent part of your product architecture. This approach saves money and time

- Security and Privacy by Default
  Make secure and privacy-friendly settings the standards in your product. Users should not need to do any extra configuration

- Zero Trust Architecture
  Always verify explicitly authenticity and correctness of identities and data in your system. Limit access rights to the least required. Do not assume that there is a "trusted perimeter", apply security hardening and verification everywhere

- Risk Transparency
  Assess and write down all security risks in your security concept. Prioritize your activities based on the inherent risk level.

But how can we integrate secure software design into our development processes and meet those requirements?

Medical regulatory bodies require us to carefully describe security requirements throughout the entire lifecycle of our software product. Only focusing on security reviews and testing would be too reactive and fail to meet these requirements. At the same time, it may frustrate the teams with externally invoked efforts that do not match their agile work paradigm, bringing in security after the software has already been developed.

Therefore, a key aspect of secure software design must be to empower the software teams to deal with the security topic along the Software Development Lifecycle in a self-determined way. One important prerequisite for this is to embrace security knowledge and responsibility within the software teams.

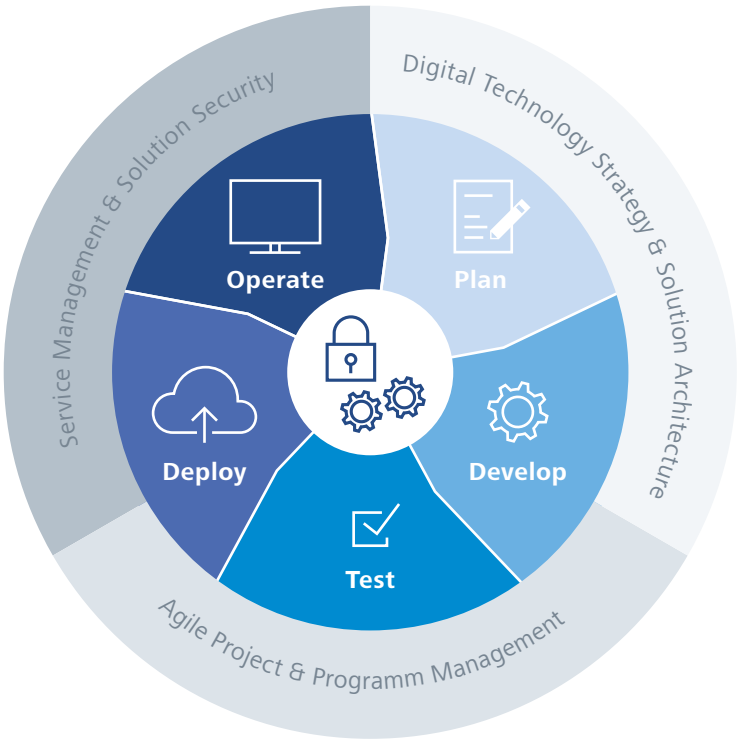

Figure 1: Software Development Life Cycle

Security Roles

Key roles in the software development team members that closely collaborate, and drive security aspects are usually Software Architects and Security Engineers. Security Engineers are software or DevOps engineers who come with a sound security mindset and who are encouraged to build a strong set of security knowledge and skills. But since Software developers need to focus on implementation, how can a diverse topic like Application Security, ranging from breaking down abstract requirements towards implementing and verifying them be comprehensively tackled on strategic and operational level?

This is where the Security Architects come into play and bridge the gap between abstract requirements, and software engineering for the development teams. This role is filled by full-time security experts who have a deep understanding on IT Security standards and procedures.

They work closely with the Security Engineer community and the software teams, break down requirements and support with the adequate tooling and methodology. The Security Architects act as translators between the strategic, business and compliance level of Application Security on the one hand, and the technical engineering level on the other hand.

On an operational level, Security Engineers and Architects work closely together on risk assessments and the identification of security requirements for the product from the initial high-level design. A key methodology applied is continuous Threat Modeling which is integrated into the agile development workflows of the Product teams.

Threat Modeling

Threat Modeling as such is a conceptual methodology that enables us to understand the key properties of a system design from a security perspective. It is important to point out that Threat Modeling is neither an automation nor a replacement for security testing.

It is, however, a format of collaboration between engineering, business, and security professionals to clarify three basic questions which can have a big impact on the security, efficiency, and cost of software development (see also Figure 2). Typically, Security Engineers, Software Architects, Security Architects and Product Owners are key participants and drivers of Threat Modeling.

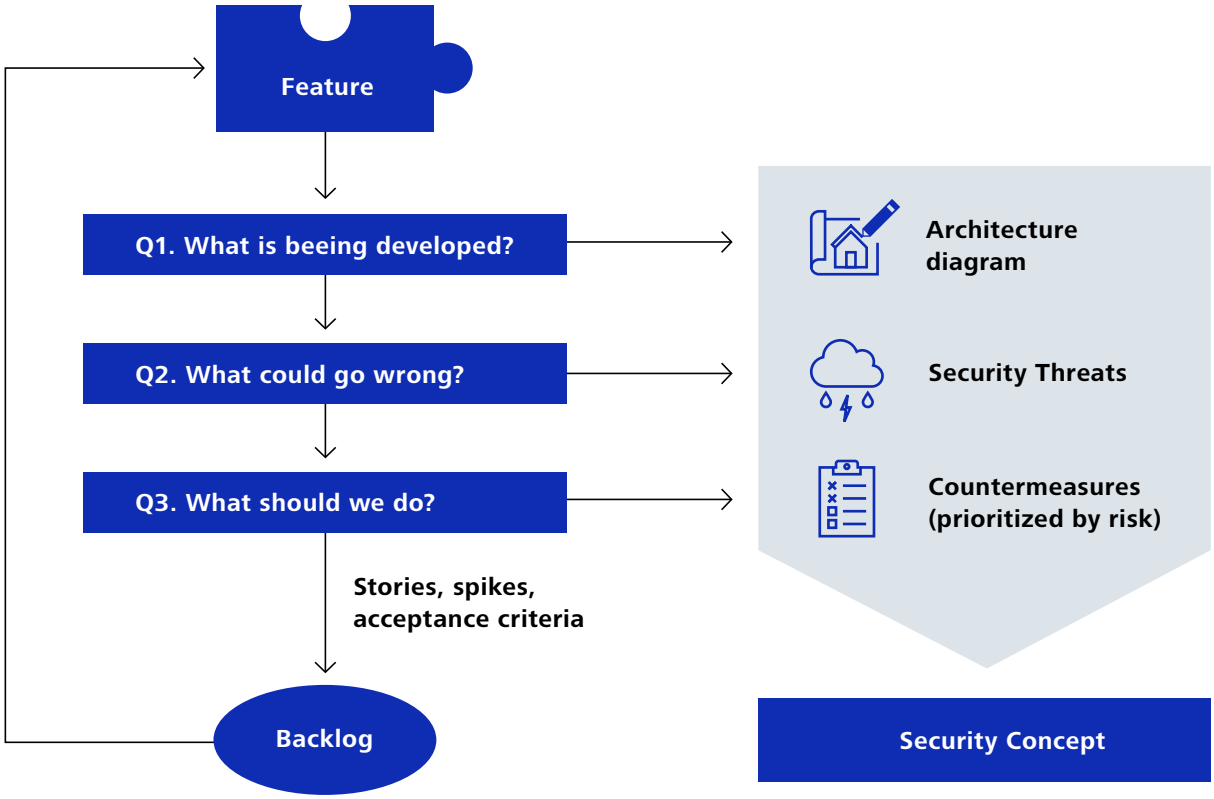

Figure 2: Thread Modeling Process

The three basic questions that the threat modeling methodology aims to answer:

Q1. What is being developed?
The architecture design is drawn in a Data Flow Diagram (DFD) which depicts what information is flowing through our system to enable which business use cases. The information processed are considered as the assets, which must be classified with a security classification based on the three dimensions confidentiality, integrity, and availability. A DFD further depicts the system components and groups them into so-called trust zones, which determine the level of exposure of the component. Components that are more publicly exposed, for instance because of their availability from public networks, are also a more prominent goal for attacks and are more probable to become victims of an attack.

For example, let's assume a product called SurgeryCloud enables a doctor to use an app to upload surgery reports to a cloud. The DFD for this simple example project could look like Figure 3.

Internet, Public Cloud, and SurgeryCloud Cloud Environment would be Trust Zones in this example, the frontend, API management, functions, and database are components, which are connected by data flow. The Surgery reports uploaded from the users would be the assets here. It would then be classified what kind of information we are processing and how to classify them from a security point of view. In this context, the surgery reports would be probably classified as PHI information, with a critical security rating and dedicated regulations likely to apply. DFDs are typically also enriched with further context information, like used protocols, TLS versions, etc.

Q2. What could go wrong?
Potential threats are identified for the architecture components, and measures are defined to counteract these threats. This is based on our assessment and can be aided by threat modeling frameworks like STRIDE. Standards, like for example the OWASP ASVS or CIS Benchmarks are also used to supply common threats and security requirements. We discuss this process with our developers and decide together which threats apply to the product and what measures need to be implemented, which can then be added to a product's backlog.

Q3. What are we going to do about it?
In this stage, countermeasures are defined to mitigate the Security threats in the given architecture. Countermeasures are often technical activities on a rather conceptual level and need to be refined by the teams into verifiable software requirements.

In the medical context, along all three stages we need to carefully understand how PHI and medical information is processed in our product design. This helps us to understand in more detail what regulatory requirements, for instance regarding encryption or audit logging, we need to apply in the concrete context. Catching up with our simple example, we would define the upload of a Surgery Report as a processing of PHI data where – depending on the target market – specific requirements towards the capturing and retention of audit logs might apply.

Besides refining the countermeasures into security requirements and adding them to the backlog as acceptance criteria, tasks or research spikes, each product team shall also maintain a security concept that wraps up all the information collected during a threat modeling. This document serves as a proof for the level of security and also assures transparency on the risks a product incurs.

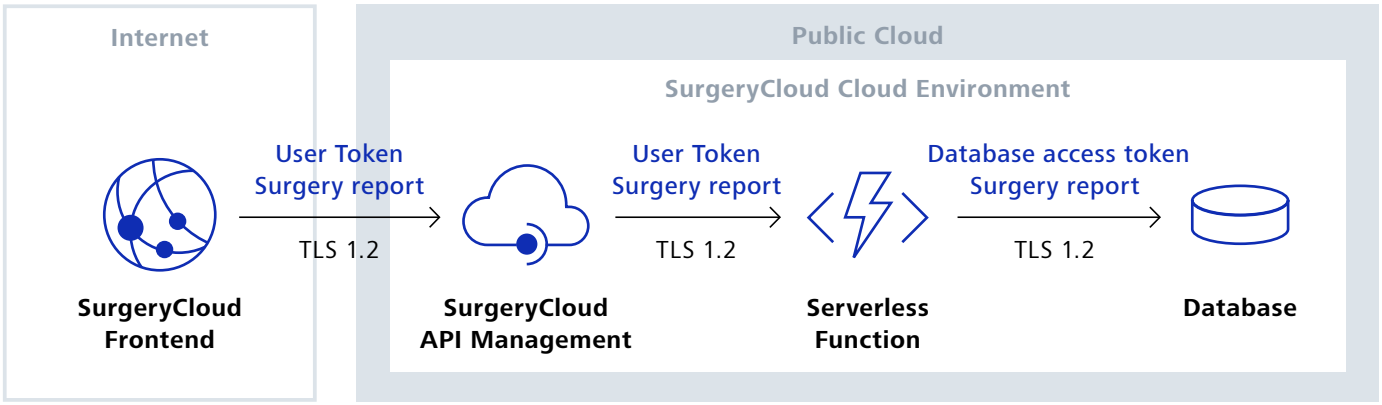

Figure 3: DFD for SurgeryCloud Example Project

Conclusion

Threat modeling makes security a part of the design phase of the Software Development Life Cycle and is more and more emerging as a key methodology to ensure the secure and compliant development of medical software. If done early in the SDLC and integrated seamlessly into the agile development procedures of the software teams, it empowers engineers to identify and remediate security requirement early on in projects and ensures that security risks are transparent to the business. However, an important prerequisite is to foster a strong culture of knowledge exchange and responsibility across the company and especially on the team level. Overarching technical Security Experts are important to foster community building, tooling, and elaboration of useful set of requirement patterns.

Altogether, threat modeling is an integral part of a Secure Software Development strategy and can greatly reduce the amount of vulnerabilities found during penetration testing and security audits.

Talk to us to find out more and visit our website to learn more about our services.

Authors

Martin Nuss
Security Architect Team
ZEISS Digital Innovation
Health & Life Science Solutions
Martin.nuss@zeiss.com

Yvi Rieck
Security Architect Team
ZEISS Digital Innovation
Health & Life Science Solutions
Yvonne.rieck@zeiss.com

Carl Zeiss Digital Innovation GmbH
Fritz-Foerster-Platz 2
01069 Dresden
Germany

Phone: +49 351 49701 – 500
info.digitalinnovation.de@zeiss.com
zeiss.com/digital-innovation