Chapter seven

Software engineering

Software Security Engineering

Security Life-cycle model:

- ✧ The Microsoft Security Development Lifecycle (SDL) is an industryleading software security process. A Microsoft-wide initiative and a mandatory policy since 2004, the SDL enabled Microsoft to embed security and privacy in its software and culture. The SDL introduces security and privacy early and throughout all phases of the development process and is without question the most widely known and used security development life-cycle model.
- ✧ Microsoft defined a collection of principles it calls Secure by Design, Secure by Default, Secure in Deployment, and Communications (SD3+C) to help determine where security efforts are needed. These are as follows:

Secure by Design

- ✧ Secure architecture, design, and structure: Developers consider security issues part of the basic architectural design of software development. They review detailed designs for possible security issues, and they design and develop mitigations for all threats.
- ✧ Threat modeling and mitigation: Threat models are created, and threat mitigations are present in all design and functional specifications.
- ✧ Elimination of vulnerabilities: No known security vulnerabilities that would present a significant risk to the anticipated use of the software remain in the code after review. This review includes the use of analysis and testing tools to eliminate classes of vulnerabilities.
- ✧ Improvements in security: Less secure legacy protocols and code are deprecated, and, where possible, users are provided with secure alternatives that are consistent with industry standards.

Secure by Default

- ✧ Least privilege: All components run with the fewest possible permissions.
- ✧ Defense in depth: Components do not rely on a single threat mitigation solution that leaves users exposed if it fails.
- ✧ Conservative default settings: The development team is aware of the attack surface for the product and minimizes it in the default configuration.
- ✧ Avoidance of risky default changes: Applications do not make any default changes to the operating system or security settings that reduce security for the host computer. In some cases, such as for security products, it is acceptable for a software program to strengthen (increase) security settings for the host computer. The most common violations of this principle are games that either open firewall ports without informing the user or instruct users to open firewall ports without informing users of possible risks.
- ✧ Less commonly used services off by default: If fewer than 80 percent of a program's users use a feature, that feature should not be activated by default. Measuring 80 percent usage in a product is often difficult because programs are designed for many different personas. It can be useful to consider whether a feature addresses a core/primary use scenario for all personas. If it does, the feature is sometimes referred to as a P1 feature.

Secure in Deployment

- Deployment guides: Prescriptive deployment guides outline how to deploy each feature of a program securely, including providing users with information that enables them to assess the security risk of activating non-default options (and thereby increasing the attack surface).
- Analysis and management tools: Security analysis and management tools enable administrators to determine and configure the optimal security level for a software release.
- Patch deployment tools: Deployment tools aid in patch deployment.

Communications

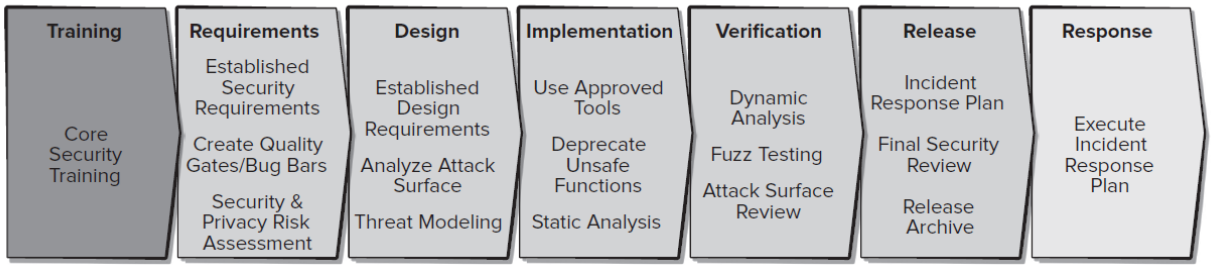
- Security response: Development teams respond promptly to reports of security vulnerabilities and communicate information about security updates.
- Community engagement: Development teams proactively engage with users to answer questions about security vulnerabilities, security updates, or changes in the security landscape.

The secure software development process model looks like the one shown in Figure 18.1.

The Microsoft SDL documentation describes what architects, designers, developers, and testers are required to do for each of the 16 recommended practices. The data that Microsoft collected after implementing the SDL shows a significant reduction in vulnerabilities, which led to a need for fewer patches, and thus a significant cost savings. We recommend that you browse the SDL website to learn more about these practices. Since the SDL was developed, there have been numerous papers, books, training, and so on, to go with the SDL model.

FIGURE 18.1 **Secure Software Development Process Model at Microsoft**

Adapted from Shunn, A., et al. Strengths in Security Solutions, Software Engineering Institute, Carnegie Mellon University, 2013. Available at http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=77878.



Security Requirements Engineering

Although security requirements are an important part of secure software development, in practice they are often neglected. When they exist, they are often an addon, copied from a generic list of security features. The requirements engineering that is needed to get a better set of security requirements seldom takes place.

Requirements engineering practice typically addresses desired user features. Therefore, attention is given to the functionality of the system from the user's perspective, but little attention is given

to what the system should not do. Users expect systems to be secure, and these assumptions need to make their way into security requirements for software systems before they are developed, not after the fact. Often the users' assumptions about security are overlooked because system features are the primary focus.

In addition to security life-cycle models, there are many process models that are specific to security requirements. These include: core security requirements artifacts, Software Cost Reduction (SCR), SQUARE (Security QUAlity Requirements Engineering), and Security Requirements Engineering Process (SREP). For the remainder of this section, we'll consider SQUARE as a representative example of security life-cycle models.

SQUARE

(1) It is a representative security requirement engineering process model, but it's important to keep in mind that if you already have a development process model, you can just pick up some of the SQUARE steps to enhance your existing model. There's no need to develop a whole new process to address security in your software development activities.

(2) The SQUARE process model provides for eliciting, categorizing, and prioritizing security requirements for software-intensive systems. Its focus is to build security concepts into the early stages of the development life cycle. It can also be used for fielded systems and those undergoing improvements and modifications.

The SQUARE process

Let's take a look at the steps.

Step 1. Agree on definitions. So that there is not semantic confusion, this step is needed as a prerequisite to security requirements engineering. On a given project, team members tend to have definitions in mind, based on their prior experience, but those definitions are often different from one another. Sources such as the Institute for Electrical and Electronics Engineers (IEEE) and the Software Engineering Body of Knowledge (SWEBOK) provide a range of definitions to select from or tailor.

Step 2. Identify assets and security goals. This step occurs at the project's organizational level and is needed to support software development. Different stakeholders usually have concerns about

different assets, and thus have different goals. For example, a stakeholder in human resources may be concerned about maintaining the confidentiality of personnel records, whereas a stakeholder in a research area may be concerned with ensuring that research project information is not accessed, modified, or stolen.

Step 3. Develop artifacts. This step is necessary to support all subsequent security requirements engineering activities. Often, organizations do not have key documents needed to support requirements definition, or they may not be up to date. This means that a lot of time may be spent backtracking to try to obtain documents, or the team will have to bring them up to date before going further.

Step 4. Perform risk assessment. This step requires an expert in risk assessment methods, the support of the stakeholders, and the support of a security requirements engineer. There are a number of risk assessment methods, but regardless of the one that you choose, the outcomes of risk assessment can help in identifying the high-priority security exposures.

Step 5. Select elicitation technique. This step becomes important when there are diverse stakeholders. A more formal elicitation technique, such as the Accelerated Requirements Method, Joint Application Design, or structured interviews, can be effective in overcoming communication issues when there are stakeholders with different cultural backgrounds. In other cases, elicitation may simply consist of sitting down with a primary stakeholder to try to understand that stakeholder's security requirements needs.

Step 6. Elicit security requirements. This step encompasses the actual elicitation process using the selected technique. Most elicitation techniques provide detailed guidance on how to perform elicitation. This builds on the artifacts that were developed in earlier steps.

Step 7. Categorize requirements. This step allows the security requirements engineer to distinguish among essential requirements, goals (desired requirements), and architectural constraints that may be present. This categorization also helps in the prioritization activity that follows.

Step 8. Prioritize requirements. This step depends on the prior step and may also involve performing a cost-benefit analysis to determine which security requirements have a high payoff relative to their cost. Of course prioritization may also depend on other consequences of security breaches, such as loss of life, loss of reputation, and loss of consumer confidence.

Step 9. Requirements inspection. This review activity can be accomplished at varying levels of formality. Once inspection is complete, the project team should have an initial set of prioritized security requirements that can be revisited as needed later in the project.

security risk Analysis

A wide variety of security risk assessment methods have been proposed. Typical examples include SEI CERT's Security Engineering Risk Analysis (SERA) method and the NIST Risk Management Framework (RMF). RMF has emerged as an approach that is widely used, providing guidelines for the users. The RMF steps for security are:

· Categorize the information system and the information processed, stored, and transmitted by that system based on an impact analysis.

· Select an initial set of baseline security controls for the information system based on the security categorization; using an organizational assessment of risk and local conditions, tailor and supplement the security control baseline as needed.

· Implement the security controls, and describe how the controls are employed within the information system and its operational environment.

· Assess the security controls using appropriate assessment procedures to determine the extent to which the controls are implemented correctly, operating as intended, and producing the desired outcome with respect to meeting the security requirements for the system.

. Authorize the information system operation based on a determination of the risk to organizational operations and assets, individuals, or other organizations (including national defense), from the operation of the information system that this risk is acceptable.

· Monitor the security controls in the information system on an ongoing basis including assessing control effectiveness, documenting changes to the system or its environment of operation, conducting security impact analyses of the associated changes, and reporting the security state of the system to designated organizational officials.