Atlantis Highlights in Computer Sciences, volume 3
21st International Scientific Workshop on Computer Science and Information Technologies (CSIT 2019)

Implementation of Secure Software Development Lifecycle in a Large Software Development Organization

Lada Gonchar
Business Configuration Development
SAP SE
Walldorf, Germany
lada.gonchar@sap.com

***Abstract*—Secure Software Development Lifecycle is an important part of developing secure software. On the one hand, such process requires a significant effort related to upskilling of developers, analysing of coding and security testing, on the other hand, generates a large amount of data on the process level (e.g. assets, dependencies, risks and mitigations) as well as on the technical level (e.g. results of static and dynamic code analysis tools). All this measure needs to be integrated in the software development process. We demonstrate how to handle this effectively by using threat modelling methodology with two different variants and generalized threat model for selected domains in the large software development organization, where we have on the one hand big variety of different application types on the other hand standardized architecture for the application development. Existing threat modelling approaches doesn't fit to SAP specific security requirements. Author proposes the generalized threat model to speed up the risk assessments and increase efficiency of security measures for ERP applications.**

Keywords*—Secure Software Development Lifecycle, threat modelling, generalized threat model, penetration testing, security validation.

I. Introduction

The large effort put into secure software development immediately raises the question of whether this investment is effective and if the effort can be invested more effectively [1].

It is known fact, that the cost to fix a bug found during test phase is costlier than one identified during implementation (Fig.1). Furthermore, bugs found in post release phase could more costly than during testing [4]. It is a reason, why it is so important to identify possible security threats in very early stage of development process. Therefore, the process is required incorporate security measures in the development lifecycle [5]. In addition, in large software development organizations such process needs to support a wide range of application types with different shipment models.

In this paper I will share my own experience in implementation of secure software development lifecycle at SAP SE, the largest European software vendor [2]. Based on this, we derive an actionable recommendation for improving secure software development.

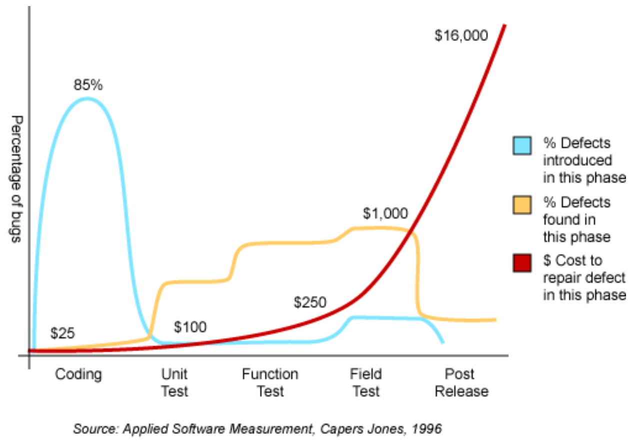

Fig. 1. Difference of costs to fix defects at each phase of software development

II. Secure Software Development Lifecycle

To ensure a secure software development, SAP follows the **SAP Secure Software Development Lifecycle** (SSDL) [2], which is inspired by Microsoft's Security Development Lifecycle [3]. Fig. 2 shows the main steps in SSDL, which is split into four phases: preparation, development, transition and utilization.

A. Preparation

The preparation phase comprises all activities that take place before the actual development start. There activities can be independent of the actual product being developed (e.g. general security awareness and regular role-specific trainings, which are mandatory for all roles contributing to the creation and maintenance of software products) or product specific (e.g. risk identification for specific product).

At the beginning of a new software development cycle, product teams conduct a security risk assessment in from of threat modelling, during which they analyse and evaluate identified risks. Threat modeling is a systematic approach to uncover security threats at design time and to support reaching a secure design [2]. It is one of the methods proposed in the Secure Software Development Lifecycle step: Security Risk Assessment. It fosters collaboration between the security expert and the architects of the scenario to drives a "think like hacker" behavior expected when striving towards a secure architecture. The security activities that the teams plan and execute later in the development lifecycle follow the results and decisions from the security risk assessment.

SAP Netweaver, to clarify and analyze dependencies. Another positive aspect was increasing security awareness in the development teams after performing threat modeling workshops, which leads to early and continuously integration of security measures in the development process.

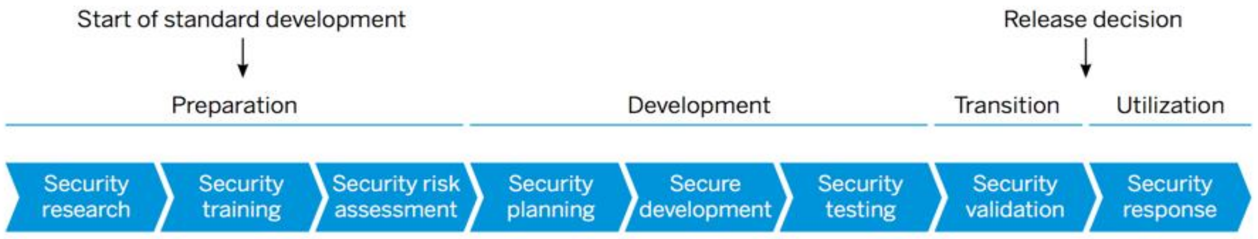

Fig. 2. SAP Secure Software Development Lifecycle (SSDL) SSDL

Developers use threat modelling in two different variants.

- **Product-level threat modelling** applies to the full product scope and architecture, comprising all parts and components, including self-developed but also open-source, third-party, freeware, outsourced, and acquired components.
- **Scenario-level threat modelling**, which is closer to the traditionally known threat-modelling approach. Product teams apply this variant for "in-depth" analysis of a particular product's components and supported scenarios. Typically, the in-depth threat modelling of selected critical scenarios often is an outcome of the in-breadth threat-modelling approach.

Challenges in the large software development organization are related to necessity to choose the right type of threat modeling and optimal number of workshops. On the one hand, it is required to have a good coverage on risk assessment across the entire organization, on the other hand, we want to avoid duplicate work in components, which can happen due to standardized development process.

From the organizational point of view especially in large development organization we can recommend following set up:

- dedicated central security team, which drives security measures and consolidates the results across the entire organization,
- security coordinators in each development area responsible for all security-relevant aspects of the product,
- security experts in each development teams supporting all security relevant activities from risk assessment to code analysis.

Recommended approach is to start with product-level threat modeling for the whole area, for example Finance and continue with scenario-level threat modelling workshops for most critical applications or scenarios, identified in step 1. Very good results were shown by involving in the threat modeling workshops cross-application experts and developers from underlying framework, for example from

Big number of conducted threat modeling workshops with continuous consolidation and analysis of results by central security team allows us to build **Generalized Threat Model (GTM)** for selected development domains. Preconditions are

comprehensive classical threat modelling workshops and several penetration tests for those domains [6].

Usually Generalized Threat Model consists of:

- assets, which should be protect,
- dependencies, which are security features provided by a component not developed by the team who build the assessed application,
- risks and corresponding mitigations.

GTM allows extremely efficient filtering for critical domains by ruling out less critical ones. By using GTM development teams can concentrate on really critical aspect of application and so speed up and increase the quality of the risk assessment process.

B. *Development*

This phase comprises the steps from planning a new product to the actual development. In particular, it covers:

- The **Planning** of Security measures, which describes the mitigation of the previously identified security risks,
- The **Secure Development** using defensive implementation strategies,
- The **Security Testing** that ensures that the planned security measures are implemented and are effective in preventing security threats.

The security test plan of a product typically contains a combination of SAST, DAST, and manual testing activities [7].

Manual testing activities are often performed by external security researchers. Here is the challenge to prepare optimal scope for testing. Our approach is to identify potential critical applications based on following criteria:

- Critical finding from threat modelling, where penetration test was recommended,
- New feature from framework implemented in the dedicated applications,

- Deviations from standard architecture.

Additional criteria is good coverage across different components, which allow us to have applications from different development units in the test.

Further improvement in the security testing will be adopting DAST tools for specific protocols used by SAP.

C. *Transition*

Security validation checks the mandatory security report against the product team's original security plan as well as against the product's security risk assessment report. In addition, security validation checks the security response plan available for the product. The security validation team also runs its own security tests.

D. *Utilization*

After the release of a product, or any extension or modification of it, the product team needs to be prepared for vulnerability reports received during use. In such a case, we must have contacts and technical skills available immediately to triage and investigate vulnerability reports and either confirm or reject the vulnerability. For a confirmed vulnerability, we must provide a security correction in time.

III. Conclusion

We have shared our experience with implementing of secure software development lifecycle and our challenges with risk assessment for complex products with different development cycles and by organizing penetration testing. Still ongoing is the research on how to improve general threat model by analysing different access paths and offering DAST tools for increasing security test coverage and better integrating in the development process.

References

[1] Empirical Research for Software Security: Foundation and Experience, 1498776418, 2017.

[2] The Secure Sofware Development Lifecycle at SAP https://www.sap.com/documents/2016/03/a248a699-627c-0010-82c7-eda71af511fa.html

[3] Microsoft's Security Development Lifecycle, https://www.microsoft.com/en-us/securityengineering/sdl

[4] Jones, Capers; Applied Software Measurement; McGraw Hill, 3rd edition 2008; ISBN 978=0- 07-150244-3.

[5] Achim D. Brucker. Bringing Security Testing To Development: How To Enable Developers To Act As Security Experts, OWASP AppSecEU 2015. https://youtu.be/LZoz4cv0MAg, https://www.brucker.ch/bibliography/abstract/talk-brucker.ea-owasp-sectest-2015.en.html

[6] Lotfi Ben Othmane, Golriz Chehrazi, Eric Bodden, Petar Tsalovski, Achim D. Brucker: Time for Addressing Software Security Issues: Prediction Models and Impacting Factors. Data Science and Engineering 2(2): 107-124 (2017)

[7] Michael Felderer, Matthias Büchler, Martin Johns, Achim D. Brucker, Ruth Breu, Alexander Pretschner: Security Testing: A Survey. Advances in Computers 101: 1-51 (2016)