Security Engineering Methods and Approaches used in SDLC

Preobrazenskaja Anna

As the use of Internet, network and thus different software increase, the importance of developing secure software increases as well. Organizations need to think whether security has to be considered at the pre or post development phase or throughout the whole software development lifecycle. It is more expensive to fix security issues once software is in production than during the development cycle. Therefore security should be addressed at all stages of the software development life cycle. The paper is aimed to provide a short introduction to the Microsoft Security Development process and relate it to different security development techniques and methods.

1 Introduction

The best approach and amount of effort, to build security into the Software Development Lifecycle (SDLC) will be different for each organization and application. An assessment of existing practices, tools, languages and frameworks, and the types of risks the software faces are a useful starting point. A gap analysis can also be undertaken to compare existing information assurance practices with those in other similar organizations. It is also common to perform a thorough application identification process and then rank them by risk, to determine where most effort should be spent.[9] Not all applications are business critical; not all applications process sensitive data; not all applications are publicly accessible. This information is used to define a software security roadmap, which can then be subsequently implemented. In this work different security techniques will be will be mapped with Microsoft Development Lifecycle stages.

2 Microsoft Security Development Lifecycle process

Microsoft’s Development Lifecycle (SDL) is a process that they have adopted for the development of software that needs to withstand security attacks [1]. It is a software development process used and proposed to reduce security issues and resolve security vulnerabilities in a timely fashion. Security analysis should be addressed through the whole development process, so Microsoft Security
Development Lifecycle efforts are grouped into seven phases: training, requirements, design, implementation, verification, release and response.

### 2.1 Training

Since a human is a weakest link in a chain, software security training is a prerequisite for implementing the SDL, and employees who are directly involved with the development of software programs must attend at least one unique security training class each year [3]. Technology evolves rapidly; therefore individuals involved with the development of software have to stay updated with the latest trends in security and privacy. Understanding software security will increase employees’ commitment to writing secure software.

It is during this phase when trust management is conducted.

### 2.2 Requirements

The Requirements phase of the SDL includes the project inception—when security and privacy at a foundational level are considered—and a cost analysis—when it is determined if development and support costs for improving security and privacy are consistent with business needs.

It's during this phase that it is needed to define the security requirements and express them in the context of a system-specific security policy. The primary goal of the security policy is to ensure the integrity, confidentiality, assurance, accountability, and availability of the system. A system that cannot be trusted will not be used [5].

According to lecture slides, there are different requirements that should be stated during this phase, namely:

- Security requirements, or in other words it should be defined what level of security is expected from the system;
- Identification requirements, extent to which a business, application, or component shall identify its externals before interacting;
- Authentication requirements, extent to which a business, application, or component shall verify the identity of its externals before interacting;
- Authorization requirements, access and usage privileges of authenticated users and client application;
- Immunity requirements, extent to which an application or component shall protect itself from infection by unauthorized undesirable programs;
• Integrity requirements, extent to which an application or component shall ensure that its data and communications are not intentionally corrupted via unauthorized creation, modification, or deletion
• Privacy requirements, extent to which a business, application, or component shall keep its sensitive data and communications private from unauthorized individuals and programs
• System maintenance security requirements, extent to which an application, component, or center shall prevent authorized modifications from accidentally defeating its security mechanisms.

Before investment of time in design and implementation, it is important to understand the costs and requirements involved in handling data with privacy considerations. Privacy risks increase development and support costs, so improving security and privacy can be consistent with business needs.

2.3 Design

The Design phase is when the plan is built for how the project will move through the rest of the SDL process. During the Design phase best practices are established to follow for this phase by way of functional and design specifications, and risk analysis is performed to identify threats and vulnerabilities in the software.

Threat modeling must be completed during project design. A team cannot build secure software unless it understands the assets the project is trying to protect, the threats and vulnerabilities introduced by the project, and details of how the project will mitigate those threats. Threat modeling applies to all products and services, all code types, and all platforms.

During this phase following risk assessment activities are conducted:
• Control analysis - current and planned controls;
• Likelihood determination - threat source motivation, capacity, nature of vulnerability, current control gap;
• Impact Analysis - mission impact, loss of confidentiality, integrity, or availability, output is an impact rating;
• Risk Determination - likelihood of exploitation, magnitude of impact, adequacy of controls, output is itemized list of risks with correlated risk level;
• Control Recommendations - formalized control recommendations list;
• Results Documentation - formalized Report which addresses threats, vulnerabilities, measures the risk, and provides recommendations for control implementation [6].

A part of this phase is building an Information System Security Risk Management (ISSRM) domain model to aid to addresses the security related issues in an IS domain [8]. The model is defined after a survey of risk
management and security related standards, security risk management methods and software engineering frameworks. The domain model supports the alignment of security modeling languages.

The CORAS framework provides a UML profile [7] for risk assessment. The proposed profile defines UML stereotypes and rules for specialized UML diagrams to support of model-based risk assessment. Asset diagram is proposed to identify assets; threat diagram captures unwanted incidents causing loss in asset value. State analysis diagram is proposed to estimate and document the frequency and consequence of the unwanted incidents, and treatments.

Misuse cases have extended the standard UML use cases to model security concerns at the early stages of software system development[8]. The misuse cases include both the graphical notation and textual representation. Misuser is defined as an actor that is willing to use the system with unfavorable intents. Initially, only threats were modeled as misuse cases. Later on, security use cases were defined as a function to protect the system assets from the identified risks. Then the misuse cases were extended with a concept of vulnerability as weakness of the system.

2.4 Implementation

During the implementation phase the documentation and tools the customer uses to make informed decisions about how to deploy the software securely are created. To this end, the Implementation phase is when development best practices are established to detect and remove security and privacy issues early in the development cycle [4].

During the phase a list of approved tools and associated security checks, such as compiler/linker options and warnings should be defined and published. The list should be regularly updated with the latest versions of the tools.

The list of banned functions should be determined, header files, newer compilers, or code scanning tools used to check code for the existence of banned functions, and then those banned functions should be replaced with safer alternatives.

2.5 Verification

During the Verification phase, it is ensured that the code meets the security and privacy tenets established in the previous phases. This is done through security and privacy testing, and a security push—which is a team-wide focus on threat model updates, code review, testing, and thorough documentation review and edit. A public release privacy review is also completed during the Verification phase.
Firstly, dynamic analysis is run-time verification of a software, leveraging tools which monitor application behavior for memory corruption, user privilege issues, and other critical security problems. It ensures software functionality works as designed.

Secondly, Fuzz Testing [4] is a specialized form of dynamic analysis that induces program failure by deliberately introducing malformed or random data to an application. It is an effective way to find potential security issues prior to release while requiring modest resource investment.

Finally, attack surface review [4] is a security practice that ensures any design or implementation changes to the system have been taken into account, and that any new attack vectors created as a result of the changes have been reviewed and mitigated including threat models.

Since trade-off analysis is a systematic examination of the advantages and disadvantages of requirements and/or design choices for a system [7] to achieve the right balance among several competing goals it is reasonable to conduct it during the verification phase. UMLsec is used to verify if the design solutions satisfy the security requirements. Design solutions that pass the verification are then evaluated using security solution design trade-off analysis.

2.6 Release

The Release phase is when the software is ready for public consumption and, perhaps more importantly, the team is ready for what happens once the software is in the hands of the user. One of the core concepts in the Release phase is planning—mapping out a plan of action, should any security or privacy vulnerabilities be discovered in the release [3]. To this end, a Final Security Review and privacy review is required prior to release.

The Incident Response Plan identifies the appropriate points of contact in case of a security emergency. It also includes security servicing plans for code inherited from other groups within the organization and for licensed third-party code. Even programs with no known vulnerabilities at the time of release can be subject to new threats that emerge over time.

The Final Security Review is a deliberate examination of all security activities which determines whether the software meets the security requirements and is secure enough for release.

Archive all pertinent information and data, including specifications, source code, binaries, private symbols, threat models, documentation, emergency response plans, and license and servicing terms for any third-party software. Archiving pertinent data and information is necessary to perform post-release servicing tasks and lower long-term costs associated with sustained software engineering.
2.7 Response

Programs with no known vulnerabilities at the time of release can be subject to new threats that emerge over time. After a software program is released, the product development team must be available to respond to any possible security vulnerabilities or privacy issues that warrant a response.

3 Summary

As we learned, Microsoft has adopted a security lifecycle for the development of software that needs to withstand malicious attack. The process encompasses of a series of security-focused activities and deliverables to each of the phases of Microsoft's software development process. Such techniques can be assigned to different Microsoft stages as trust management, security trade-off analysis, role-based access control, security modeling, and many others.

Knowing how mighty Microsoft is and how popular its products are I can conclude that it indicates that the SDL is effective at reducing the incidence of security vulnerabilities.

References