CERT'S PODCASTS: SECURITY FOR BUSINESS LEADERS: SHOW NOTES

Identifying Software Security Requirements Early, Not After the Fact

Key Message: During requirements engineering, software engineers need to think deeply about (and document) how software should behave when under attack.

Executive Summary

Capturing security requirements during the requirements engineering phase is the most cost effective way to ensure that the final software operates with the expected level of security. This approach is strongly recommended in addition to today's current practice of deploying operational security controls such as firewalls, passwords, and frequent patching with the hopes of protecting vulnerable software.

In this podcast, Nancy Mead, the leader of CERT's research efforts in security requirements engineering, discusses why it's important to identify software security requirements early in the software development life cycle (SDLC) and practices for getting started.

PART 1: EARLY IDENTIFICATION REDUCES TOTAL COST

Justifying Resources to Develop Security Requirements

Some studies show that it costs 10 to 200 times more to fix a requirements problem once a system is in operation.

Reworking requirements defects can account for 40-50% of total software project effort.

What we're doing today clearly isn't working; vulnerabilities in critical operational systems are discovered daily.

Addressing security early in the SDLC can result in:

- maintaining reputation
- satisfying client concerns
- reducing effort spent in issuing and installing patches and responding to security break-ins

Architectural decisions made in the absence of security requirements make it difficult to add security later on.

On the Other Hand

Often the project manager responsible for developing the software is not the same one who is responsible for operating the software. In other words, no single role has responsibility for reducing total life cycle cost of ownership.

How do you make a compelling argument in this situation – where benefits are realized after the software has been developed?

Organizational reputation is a key factor. A fielded product with vulnerabilities is going to affect reputation. Leadership and management support are key for making good investment decisions.

Well-Known Security Controls Are Not Enough

Scanning code for vulnerabilities, doing regular penetration testing, using strong passwords, and installing firewalls are certainly necessary but not sufficient to protect software from attack.

We do these things today but still suffer software security breaches.

Adding on mechanisms such as encryption is a fine general purpose solution for one class of problem but practices such as these are not tailored to the system under development.

These types of well-known security controls are typically tacked on after the fact, once the system is already in operation. They tend to be reactive in nature. We just can't keep using only this approach.

PART 2: HOW ARE SECURITY REQUIREMENTS DIFFERENT?

What Makes Security Requirements Different from Other Software Requirements

Most software requirements address what the software should do (functional requirements), as viewed by a typical user (end-user requirements).

Typical methods for capturing requirements don't do a very good job of addressing what are sometimes referred to as non-functional requirements such as quality, maintainability, availability – and security.

Security requirements tend to get overlooked in most requirements engineering methods. Security isn't a user-oriented feature.

One way to begin to tease out security requirements is to think like an attacker. Most software engineers, developers, and users don't think about what should happen when the system is under attack.

What Developers Need to Know

It turns out that eliciting security requirements does not require a great deal of extra skill beyond those that are already present on the team. The best way to get started is to have a software engineering team member with security expertise.

Software engineers, for example, can be very good at developing use cases (how the software will be used) while security engineers are typically very good at developing misuse and abuse cases (how the software should function in unexpected circumstances).

PART 3: GETTING STARTED

Start with a Sound Foundation

First, understand your current software engineering process and practices. This is the best foundation for adding in a security requirements engineering process.

That said, if you don't have a sound software development process, performing security requirements engineering is going to be very difficult – and will not be sustainable from project to project.

Use a Defined Security Requirements Engineering Process

One example is SQUARE (Security Quality Requirements Engineering) – a 9-step process for defining security requirements.

Benchmark what other organizations are doing and select and tailor an approach that fits with your organization and software development process.

Make sure there is a common understanding of widely used terms, such as availability, across your development team.

Document your security goals.

Perform a security risk analysis as a companion to other risk analyses, such as those for cost and schedule.

Make sure that the elicitation technique you intend to use will capture both functional and non-functional requirements, such as security.

Perform cost-benefit analysis on candidate security requirements. Funding is typically not sufficient to address all of these.

Resources

Allen, Julia; Barnum, Sean; Ellison, Robert; McGraw, Gary; Mead, Nancy. *Software Security Engineering: A Guide for Project Managers*, Addison-Wesley, 2008.

The Department of Homeland Security Software Assurance Program's Build Security In web site, specifically the Requirements Engineering content area

SEI web site (search under Library on "SQUARE")

CERT SQUARE web site

CERT podcast: How to Start a Secure Software Development Program

CERT podcast: Building More Secure Software