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

How to Start a Secure Software Development Program

Key Message: Software security is accomplished by thinking like an attacker and integrating security practices into your software development life cycle.

Executive Summary

Today's businesses are increasingly exposed to software-induced business risks due to their reliance on software-intensive IT systems and critical applications. Many organizations start with tools and training but these approaches are sub-optimal. Moreover, there are two key practices that specifically target software design flaws and coding bugs – the source of most security defects and vulnerabilities. These are architectural risk analysis and source code analysis, respectively.

In this podcast, Gary McGraw, Chief Technology Officer for Cigital and the editor of Addison Wesley's Software Security Series, discusses why organizations are paying more attention to software security and outlines approaches for initiating a software security program.

PART 1: THE EVOLUTION OF SECURE SOFTWARE DEVELOPMENT

Background

Gary McGraw and John Viega wrote *Building Secure Software: How to Avoid Security Problems the Right Way*, in part based on their observation that there was very little written about how to develop secure software. This was also informed by Gary's early work in finding security holes in Java.

Interest in software security has moved from evangelism and awareness (5 years ago) to actually trying to do something about insecure software. Today's actions include:

- integrating security best practices into the SDLC
- defining metrics
- establishing governance programs at the executive level
- leadership from the financial services industry in terms of software security practice adoption

Other industries starting to take action include software vendors (Microsoft as a case in point), embedded systems and cell phone technology providers (for example, Qualcomm), and the hospitality industry.

Motivation

Software security includes actions taken to make software behave as expected under intentional malicious attack.

Most organizations don't realize how much software they build and the risks that arise as a result. Software risk = business risk.

In the financial services sector, one role of a Chief Risk Officer is figuring out how to deal with software risk, including software security risk. Realizing the business's exposure to software-induced risk has raised the profile.

Developing Software with Security in Mind

The biggest challenge when developing secure software is being able to think like an attacker.

Most developers are quite capable of addressing what it takes to make software behave under normal conditions (solid requirements engineering, design, analysis, and use of coding standards).

But when faced with actions that an attacker might take, developers will often say "Well, nobody would ever do that. That's against the rules."

Software security best practices (Gary's touchpoints) are designed to aid developers in addressing how attackers think and act. They are inserted throughout the SDLC. These practices are process-neutral, that is, they don't rely on a particular software method or process.

PART 2: KEY PRACTICES FOR NEW SOFTWARE; COTS AND LEGACY SOFTWARE

Key Practices When Developing Software

Of the seven touchpoints described in *Software Security: Building Security In* and *Software Security Engineering: A Guide for Project Managers*, the two that are most important are:

- source code analysis with a static analysis tool
- architectural risk analysis

These are most important because software security flaws occur in two ways: bugs created during coding and flaws created during design. Bugs and flaws are generally divided 50/50.

These two practices are essential given an objective to reduce security-related defects by minimizing the occurrence of bugs and flaws.

Roles and Responsibilities

A key enabler of software security is to make sure it is someone's actual job. Traditionally software developers think this is IT security's job (firewalls, anti-virus), and IT security thinks that software developers provide buggy software and expect them to operate it securely.

When this happens, no one is responsible.

Considerations for COTS (Commercial-Off-The-Shelf) and Legacy Software

Business consumers are starting to demand more secure software products from their vendors and suppliers. Assurance cases are one of the practices that aid in providing evidence of more secure software.

Service level agreements and other legal constructs can be developed to include acceptance criteria for COTS and legacy software.

PART 3: TWO WAYS TO GET STARTED; TWO TO AVOID

Top Down

One approach is effective for organizations that have strong centralized IT leadership with the power and budget to drive organizational and cultural change. Steps to take for this type of program include:

- developing a plan with a two to three year horizon that defines current state, desired state, and how to get there.
- training lots of people
- building a web-based portal to serve as a central place for learning, cataloging secure code for others to use, and

describing how bugs and exploits work
- buying the right tools and integrating them into code reviews and testing processes
- integrating software security practices into all phases of the SDLC

Stove Piped

A second approach works well when you have strong centralized IT leadership along with separate vertical business units that have no direct reporting relationships to IT.

The key step here is to perform a portfolio risk analysis of all software applications to determine which ones are the most important and carry the most risk. This helps prioritize software security investments.

Tool-Centered (less optimal)

Many organizations tackle software security by buying as many tools as they can get their hands on (web application security, static analysis). Buying and installing tools in the absence of a risk-based plan or process is not effective.

Training-Centered (also less optimal)

Another approach that is sometimes used is to train your development staff based on advice from your security "geeks." But training only works when it is integrated into the SDLC – not as a standalone endeavor.

More details about these four approaches are available in Gary's article "Software Security Strategies."

Resources

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

Addison-Wesley Software Security Series

The Department of Homeland Security Software Assurance Program's Build Security In web site

McGraw, Gary. "Software Security Strategies: Four ways to kick off your organization's software security initiative in the New Year." Dark Reading, September 18, 2008.

CERT podcast: Building More Secure Software

CERT podcast: Identifying Software Security Requirements Early, Not After the Fact

Gary McGraw's Silver Bullet Podcast Series

Gary McGraw's Justice League blog

Additional Gary McGraw articles and publications