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

Building More Secure Software

Key Message: Software security is about building better, more defect-free software to reduce vulnerabilities that are targeted by attackers.

Executive Summary

In addition to computer, network, IT, and information security, software security (which includes application security) is starting to emerge as the next area to tackle. Software security means building better, defect-free software that is more able to resist, tolerate, and recover from attacks. As the security community deploys more effective solutions to address IT and information security vulnerabilities, attackers are targeting vulnerable application software with much greater frequency.

In this podcast, Julia Allen, a senior researcher with CERT, discusses why business leaders need to start paying attention to software security. Julia, along with several CERT and Cigital authors, has written a new book in the Addison-Wesley SEI/CERT and Software Security Series titled *Software Security Engineering: A Guide for Project Managers*. This book and the Department of Homeland Security Software Assurance Program's Build Security In web site serve as the basis for this conversation.

PART 1: SOFTWARE SECURITY IS JUST GOOD BUSINESS

Defining Software Security

Typically, when organizations think about security, they think about physical, IT, computer, network, and information security; or they think about protecting sensitive and personal information.

Software security is:

- building better, defect-free software to reduce the number of software vulnerabilities that show up in operational production systems
- developing software to be more resistant to attack and, if attacked, the software is better able to tolerate the attack and recover quickly
- promoting more preventive and proactive measures, by addressing security issues much earlier in the life cycle, where they are often first introduced

Software that is built with security in mind is less vulnerable to attack and a bit more bulletproof.

Attackers Are Getting Smarter

Attackers:

- are getting much smarter and more sophisticated in their approaches. There is a robust underground economy where information is bought and sold.
- are moving their focus from networks and IT infrastructures to applications and information. For example, web-facing applications serve as one primary gateway to desirable information.
- only require one entry point for success, whereas software and IT developers need to protect all entry points.

Software Security Is Good Business

The marketplace is starting to demand more secure software products.

The total cost of ownership for software can be upwards of 50-80% during the operations and maintenance phases, due to poor software quality.

Upwards of 50% of software vulnerabilities are design flaws that could have been detected much earlier in the life cycle.

It can be anywhere from 100 to 1000 times more cost and schedule effective to identify a software defect earlier in the life cycle versus finding it during operations.

We just can't keep up by addressing software security solely as an operational concern.

Why Is Software So Complex?

Software complexity stems from:

- customers wanting new features, new functions, and new services faster
- the growth in the amount of software
- Internet connectivity that facilitates global software development
- new software being added to existing systems
- the increasing use of third party software

Addressing security in the face of this increasing complexity is a daunting undertaking.

PART 2: DEVELOP SOFTWARE WITH A SECURITY MIND-SET

How To Think About Software Security

Tackle security as a software development life cycle issue:

- Understand that software security is not just an operational IT issue and an add-on or afterthought - it is a software engineering issue.
- Address security during acquisition, requirements specification, design, architecture – and all the way through implementation, test, and deployment.
- Think about security in the same way that you think about software performance and reliability.

Think like an attacker:

- Involve all roles – software project managers, architects, designers, software engineers.
- Think about what the software should NOT do (not just what it should do) and how it should behave when under attack.

Address software security as a risk management issue, assessing risk continuously during each life cycle phase. Risks will change over time.

Some Useful Software Security Practices

First of all, integrate software security practices into your organization's software development life cycle. Don't make it something new or distinct from your normal process.

Examples of good practices include:

- defining misuse and abuse cases, thinking about how new features could be unintentionally misused or

intentionally abused by an attacker. An example is challenging the assumption that the interface between a user-facing web server and a database server can be trusted.
- capturing attack patterns which describe a class of vulnerability, how that vulnerability can be exploited, and the level of attacker skill required. Attack patterns can be used to better understand weaknesses during all development phases.
- secure coding practices, code scanning, and code analysis
- security testing to include white box testing, black box testing, threat modeling, and penetration testing.

Getting Started

Useful first steps include:

- evaluating the skills and competencies of your software development team. Add security expertise to the team where needed.
- inserting security practices as early in the life cycle as possible for the greatest return on investment. Keep in mind that early life cycle software security practices are the least mature.

That said, most organizations today start with secure coding practices along with code analysis, peer reviews, and lower-level testing. These practices are in broader use and thus more mature.

For any improvement initiative, we recommend:

- understanding what you are trying to do and why, and what you want to accomplish. Make sure the incentives are clear.
- obtaining active, visible buy-in from executive sponsors for a sustainable initiative
- selecting early pilot opportunities that can demonstrate early results (quick wins)
- making sure that people understand this is going to take time
- communicate, communicate, communicate
- provide ongoing awareness training and education

Tackling the tough issue of legacy systems and third-party software, while challenging, can result in high payoff.

Resources

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

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

CERT podcast: How to Start a Secure Software Development Program

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