Editors: Brian Chess, chess@hp.com
Brad Arkin, barkin@adobe.com

Software Security in Practice

Brian Chess
Fortify Software

Brad Arkin
Adobe Systems

This department is about building software with security in mind. Since it began in 2004, it has focused on the kinds of activities that constitute a secure development life cycle. As of today, we're broadening that charter to include all the essential ingredients of a sustained software security initiative. Instead of focusing on one turn of the crank that yields one new piece of software, we'll consider the ongoing organizational commitments necessary to facilitate secure software development.

We're thinking big: a lot of programmers making a lot of software over a long period. Most such organizations are commercial or institutional, making software to either sell (such as independent software vendors like Microsoft or Adobe) or use themselves (such as banks or government agencies). Other kinds of organizations have software security concerns too—single developers making mobile applications or groups of volunteers making open source software. But in our observation, most concentrated security efforts involve a big group trying to march to a single drummer.

We gravitate toward the empirical and away from the theoretical. Software is infinitely malleable, and in theory there are few hard-and-fast rules about how to make it. But software security problems are real and are being addressed every day. In practice, we note many common activities across many organizations. We're going to focus on those activities and dive into the nitty-gritty reality behind modern software security efforts.

Security in the Development Life Cycle: Necessary but Not Sufficient

For us to have any hope of creating secure software, security must be part of software development. Figure 1 appeared in the very first article in this department.[1] Adapted from Gary McGraw's book *Software Security*,[2] it overlays security-focused activities onto the major software development phases. (This illustration implies a waterfall development model, but you could create a similar overlay for just about any software development methodology.) Gary calls these activities *software security touchpoints*. Microsoft's Security Development Lifecycle (SDL; www.microsoft.com/security/sdl) and Adobe's Secure Product Lifecycle (SPLC; www.adobe.com/security/splc) define a similar set of activities specifically to overlay onto these orga-

 1540-7993/11/$26.00 © 2011 IEEE COPUBLISHED BY THE IEEE COMPUTER AND RELIABILITY SOCIETIES

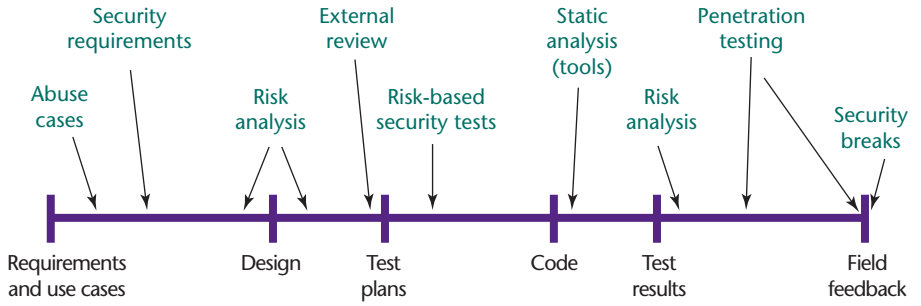

Figure 1. Security activities as part of a development life cycle.[1] The story starts with requirements and use cases and ends with software in the field. Although this is a reasonable way to think about the journey of a single program, it hides much of the effort required to create a repeatable process, and it ignores topics such as compliance and training.

nizations' software development approaches.

But in practice, big organizations' software security teams end up doing a lot of things that aren't part of the life cycle for any particular application. Besides working on the assembly line, they must build and maintain the factory. Their concerns include these:

- *Building the security team*. Attracting, recruiting, and retaining security talent is a challenge. The white-hot market for software security skills means that teams frequently must "build" security personnel from nonsecurity technical resources.
- *Maintaining legacy code*. Security threats and offensive techniques are forever marching forward. Maintaining a consistent level of security for a stable feature and functionality set in a cost-conscious environment is a serious challenge for most organizations.
- *Gaining the organization's support (budget and respect)*. Having a great plan is only a small part of a real-world solution. Achieving the right priority ranking for security alongside all the other business issues competing for attention is critical for any nascent security program.
- *Establishing an education and training program*. Organizations must arm their employees with the skills and knowledge required to support the software security initiative.
- *Governance*. Consistency and repeatability across many teams and concurrent development projects require checkpoints and enforcement opportunities.
- *Establishing standards and metrics*. How much is enough? How does a team know when it's done (for now)?
- *Interfacing with third parties who feed the software supply chain*. Interdependencies in software development are growing rapidly. Does an application's open source component meet the security standards set for the internally produced code? Merely doing a great job building secure code in-house isn't enough.
- *Handling breaches and incidents*. No software is perfect or free of vulnerabilities. The ability to respond appropriately to security events that happen in production or are identified after a product has shipped is just as important as the work to prevent those vulnerabilities from being introduced during development.
- *Managing piles of knowledge about vulnerabilities, attacks, attackers, and more*. What's relevant and what's just fear, uncertainty, and doubt? Filtering the information security fire hose for actionable intelligence requires conscious thought and resources.
- *Tooling*. Effective teams use tools to automate security tasks during software development, the moment something becomes repeatable. Build-versus-buy decisions and the art of determining how much is "good enough" are both part of maintaining an effective set of tools.
- *Interfacing with the wider software security community*. Large organizations don't operate in a vacuum. Their decisions and public comments influence the behavior of people in the white-hat security community as well as black-hat bad guys. PR, marketing, conference participation, and one-on-one relationships might be critical, depending on the software producer's profile and types of offerings.
- *Wrangling compliance requirements and the auditors who love them*. Regulatory compliance seeks to drive software producers toward building better software, but too often the activities might feel orthogonal to what really works based on empirical evidence. When should an organization just "check the box," and when should it seize on a compliance activity as an opportunity to drive real, effective change?
- *Building feedback loops for continuous improvement*. Security is a moving target; any effective software security capability must be dedicated to constant reinvention.

This list is just a taste of the broader concerns confronted by any organization attempting to secure software in a real-world environment. A software engineering process won't yield robust, safe software unless it addresses these issues (and the many others we didn't enumerate).

Table 1. The BSIMM software security framework.*

Governance	Intelligence	Secure software development life cycle touchpoints	Deployment
Strategy and metrics	Attack models	Architecture analysis	Penetration testing
Compliance and policy	Security features and design	Code review	Software environment
Training	Standards and requirements	Security testing	Configuration management and vulnerability management

* The framework organizes 12 practices into four domains. The activities most often associated with the development life cycle take up only one domain (secure software development life cycle touchpoints).

BSIMM: Modeling a Software Security Initiative

The Building Security In Maturity Model (BSIMM; www.bsimm.com) is a study of real-world software security initiatives. (Full disclosure: Brian is one of the study's authors, and Brad serves on the BSIMM advisory board.) One motivation for the study was to take a step back from the project-level view of software security activities and look more closely at the behaviors of organizations that routinely leverage these activities while building software. The result is a framework that helps categorize and describe the "glue" in an organization that allows software security to succeed at the project level.

The latest BSIMM results are derived from 30 major software security initiatives. These 30 organizations display considerable diversity in the structure of their software security groups (SSGs), the SSGs' activities, and the organizations' overriding objectives. BSIMM provides a model and common language to describe similar activities observed across what are otherwise diverse software-producing environments.

The verticals represented in BSIMM include (with some overlap): financial services (12 organizations), independent software vendors (7), technology (7), healthcare (2), insurance (2), energy (2), and the media (2). The participants willing to be identified by name are Adobe, Aon, Bank of America, Capital One, EMC, Google, Intel, Intuit, Microsoft, Nokia, Qualcomm, Sallie Mae, Standard Life, the Society for Worldwide Interbank Financial Telecommunication, Symantec, Telecom Italia, the Depository Trust & Clearing Corporation, Thomson Reuters, VMware, and Wells Fargo.

Table 1 shows BSIMM's software security framework. It breaks a software security initiative down into four domains:

- *Governance*—practices that help organize, manage, and measure the initiative. Staff development is also a central governance practice.
- *Intelligence*—practices resulting in collections of corporate knowledge used to carry out software security activities throughout the organization. Collections include both proactive security guidance and organizational threat modeling.
- *Secure software development life cycle touchpoints*—practices associated with analysis and assurance of particular software development artifacts and processes. All software security methodologies include these practices.
- *Deployment*—practices that interface with traditional network security and software maintenance organizations. Software configuration, maintenance, and other environment issues directly affect software security.

Each domain has three practices, each with a set of activities (not shown in Table 1). Activities are grouped in three levels, with succeeding levels generally corresponding to more advanced or more difficult activities. The 12 practices comprise 109 activities. No organization in the study performs all 109 activities.

Here's an example activity from the security features and design (SFD) practice:

> ***SFD 2.1: Build secure-by-design middleware frameworks/common libraries.*** The SSG takes a proactive role in software design by building or providing pointers to secure-by-design middleware frameworks or common libraries. In addition to teaching by example, this middleware aids architecture analysis and code review because the building blocks make it easier to spot errors. For example, the SSG could modify a popular Web framework such as Struts to make it easy to meet input validation requirements. Eventually the SSG can tailor code review rules specifically for the components it offers. (See Code Review 3.1: Use automated tools with tailored rules.)

Besides gathering information about the organizations' software security activities, BSIMM gives information about the organizations' sizes and shapes. All 30 BSIMM participants have an in-

Welcome New Department Editors

Brian Chess is the founder and chief scientist of Fortify Software, now a Hewlett-Packard company. Chess has a BS, an MS, and a PhD in computer engineering, all from the University of California, Santa Cruz. He's the coauthor of *Secure Programming with Static Analysis* (Addison Wesley, 2007). Contact him at chess@hp.com.

Brad Arkin is the senior director of product security and privacy for Adobe Systems. Arkin has a BS in computer science and mathematics from the College of William and Mary, an MS in computer science from George Washington University, and an MBA from the Columbia Business School and London Business School. Contact him at barkin@adobe.com.

ternal group devoted to software security—the SSG. On average, an SSG has 21.9 core people (smallest, 0.5; largest, 100; median, 13) and 39.7 "satellite" people (smallest 0, largest 300, median 11). Satellite people include developers, architects, and people directly engaged in and promoting software security. The average number of developers is 5,061 (smallest, 40; largest, 30,000; median, 3,000), yielding an average percentage of SSG to development of just over 1 percent (largest, 2.6 percent; smallest, 0.05 percent).

Although none of the 30 SSGs have exactly the same structure, some themes are common. Some SSGs are highly distributed; others are centralized and policy oriented. There are several common subgroups: people dedicated to policy or strategy and metrics; internal "services" groups that (often separately) cover tools, penetration testing, and middleware development or shepherding; incident response groups; groups that develop and deliver training; externally facing marketing or communications groups; and vendor-control groups.

Over the past decade, software security has matured from a niche topic studied by academics and precious few in industry to a vital part of how software is built. This department's focus tracks well to the leading edge of the state of the practice of software security. Now that "building security in" is well understood for greenfield individual software development projects, we focus on the challenge of creating a fertile environment for software security to flourish across the breadth of a large organization.

We invite you to share your real-world experiences with establishing, managing, and evolving software security initiatives at your organizations. (There are also a few juicy topics we plan to save for ourselves!) Throughout all this, we plan to leverage BSIMM to help map lessons learned in a specific environment to a broad array of software security initiatives. We look forward to sharing what we've learned. Please join us! ☐

References

1. G. McGraw, "Software Security," *IEEE Security & Privacy*, vol. 2, no. 2, 2004, pp. 80–83.
2. G. McGraw, *Software Security: Building Security In*, Addison-Wesley Professional, 2006.