

Adopting an Enterprise Software Security Framework

Most organizations no longer take for granted that their deployed applications are secure. But even after conducting penetration tests, network and hosting security personnel spend considerable time chasing incidents. Your organization

some application security guidelines or even buying a static analysis tool—essentially, picking the low-hanging fruit.

Although it sounds compelling, avoid charging off to win this easy battle. If your organization is large, you don't need to be reminded of what role politics plays. At the director level, headcount, budget, and timelines are the system of currency, and demanding that development teams adhere to guidelines requiring development they haven't budgeted for, or imposing a tool that spits out vulnerabilities for them to fix prior to release, can quickly send software security efforts into political deficit.

To use a war analogy, each of the chief information officer's majors must understand their role in software security prior to going into battle. To win, each major will have to take on at least a small amount of responsibility for software security, but the most crucial aspect of success is for each of them to know his or her responsibility and when to collaborate.

Put simply, without executive sponsorship, a unified understanding of roles, responsibilities, and a vision for software security, the effort will sink quickly into political struggle or inaction.

Creating a new group

Some organizations respond to the software security problem by creating a group to address it. Headcount and attention are necessary, but it's a mistake to place this headcount on an island by itself. It's an even bigger mistake to use network security folks to create a software security capability—they just don't understand software well enough.

JOHN STEVEN
Cigital

might be one of the many that have realized the "secure the perimeter" approach doesn't stem the tide of incidents because the software it's building and buying doesn't resist attack.

Painfully aware of the problem, savvy organizations have grappled with how to build security into their software applications for a few years now. Even the best efforts have met considerable resistance because the problem is mostly organizational and cultural, not technical—although plenty of technical hurdles exist as well.

Unfortunately, software security might be new to your organization's appointed "application security" czar—if one even exists. Even knowing where to start often proves a serious challenge. The first step toward establishing an enterprise-wide software security initiative is to assess the organization's current software development and security strengths and weaknesses. Yes, this applies to software built in-house, outsourced, purchased off-the-shelf, or integrated as part of a vendor "solution."

As an exercise, ask yourself the first question in my imaginary software security assessment: "How much software does my organization purchase compared to how much it builds in-house?" If the overwhelming majority of deployed software is

outsourced, software security looks a lot more like outsourced assurance than it does building security in! Most organizations do quite a bit of both, so we'll have to solve both problems.

Common pitfalls

Whether tackling the problem formally or informally, top-down or bottom-up, organizations hit the same roadblocks as they prepare to build and buy more secure applications. How each organization overcomes these roadblocks depends a great deal on its strengths and weaknesses: no one-size-fits-all approach exists. Just knowing some of the landmines might help you avoid them, though. Let's look at some of the most common ones.

Lack of enterprise-wide software security goals, vision

It bears repeating: the first hurdle for software security is cultural. It's about how software resists attack, not how well you protect the environment in which the software is deployed. Organizations are beginning to absorb this concept, but they don't know exactly what to do about it. Their first reaction is usually to throw money and one of their getters at it. He or she might make some progress initially by defining

Software security resources must be placed into development teams and seen as advocates for security, integration, and overcoming development roadblocks.

Software security best-practices nonexistent

Security analysts won't be much more effective than penetration testing tools if they don't know what to look for when they analyze software architecture and code. Likewise, levying unpublished security demands on developers is nonproductive and breeds an us versus them conflict between developers and security.

Instead, build technology-specific prescriptive guidance for developers. If the guidance doesn't explain exactly what to do and how to do it, it's not specific enough. Specific guidance removes the guesswork from the developer's mind and solves the problem of consistency between security analysts.

Software risk doesn't support decision-making

Although most organizations view critical security risks as having the utmost importance, project managers constantly struggle to apply risk-management techniques. The first reason for this is a lack of visibility. Even if a technical vulnerability is identified, analysts often don't fully understand its probability and impact. Rarely does an organization use a risk-management framework to consistently calculate a risk's impact at the project-management or portfolio level.

Establish a common risk framework as part of governance efforts to gain business owners' understanding and respect if you want the organization to choose security risk over time-to-market or if you need additional capital when making release decisions.

Tools as the answer

Companies often believe that an authentication, session management,

data encryption, or similar product protects their software completely. Although they serve as lynchpins of an organization's software security proposition, most organizations have a weak adoption of these tools at best. What's worse is that these technologies are often deployed without being properly vetted. Not only do the products themselves possess vulnerabilities, but the organization's development teams weren't consulted to help with deployment, making integration difficult if not infeasible. Even if adoption of these tools were complete, they would not in and of themselves assure that an application could resist attack. Too often, architecture review is reduced to a checklist: "Did you integrate with our single sign-on and directory service tools?" "Yes? Then you're done." It's no wonder these applications still possess exploitable architectural flaws.

Penetration testing and static analysis tools aren't panaceas either. These tools help people find vulnerabilities, but there's a lot more to building security into software applications than running these tools, as we'll see.

Framing the solution

An Enterprise Software Security Framework (ESSF) is a new way of thinking about software security more completely at the enterprise level, targeting the problem directly without demands for massive headcount, role changes, or turning an IT shop upside down to prioritize security ahead of supporting the business that funds it. ESSFs align the necessary people, know-how, tech-

software. Because every organization possesses different strengths and weaknesses, and, most important, faces different risks as a result of using software, ESSFs will differ across organizations. Although I can't say what elements the optimal ESSF in your organization will look like, I can present properties that all good ESSFs will possess.

"Who, what, when" structure

To align each group's role in achieving secure software, an organization's ESSF should possess a "who, what, when" structure—that is, the framework should describe what activities each role is responsible for and at what point the activity should be conducted. Because building security into applications requires the collaboration of a wide variety of disciplines, the framework should include roles beyond the security analysts and application development teams. Figure 1 shows column headings under which an ESSF might list each role's responsibility.

You might not recognize some of the group names in the figure. One organization's infrastructure group is another's shared services or architecture office, or something else entirely, and that's okay. Another subtlety involves reporting relationships—although they're important, don't get wound up in them when defining an ESSF. Focus on who needs to do what.

Figure 2 shows a partial enumeration of activities for which a particular role is responsible. Each group further defines how they accomplish each of their framework activities.

Patience: it will take at least three to five years to create a working, evolving software security machine.

nologies, and software development activities to achieve more secure

For example, the business owner of development might decide to build a

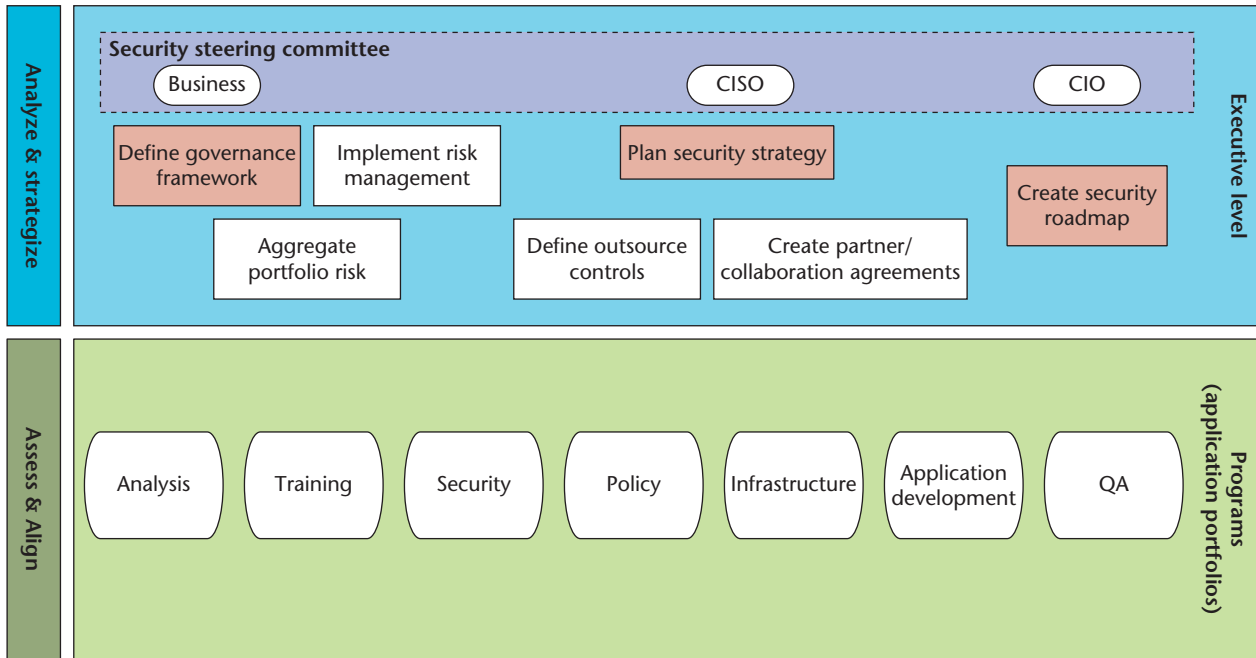


Figure 1. Role responsibilities: who. The pink boxes represent each role’s first steps.

handbook to walk developers step-wise through the process of programming securely. It’s just as likely this wouldn’t get traction, though, so the ESSF could mandate training for developers before they’re unleashed into the development organization. It’s unclear what activities will comprise your organization’s ESSF, but here are a few gotchas to avoid:

- Don’t place technologies or products, such as “single sign-on,” in the framework’s boxes.
- Don’t demand teams to begin conducting every activity on day one. Slowly introduce the simplest activities first, then iterate.
- Avoid activities that produce unverifiable artifacts or results, such as “collaborate with security here.”

Remember, the framework’s primary goal is to align people and their responsibilities, so keep the visuals about who does what activities when.

Focus on resisting attack, not including

security features

Security is an emergent property of an application, not just an amalgam of security features. In an attempt to simplify the problem and make initial strides, organizations often get stuck in a feature-centric mode: they tell themselves, “If we just encrypt our HTTP connections and authenticate users, we’re doing enough.” Thinking about how to leverage the security features of toolkits, languages, and application servers within an application is good and necessary—it just isn’t sufficient. To be successful, the philosophy of “resisting attack” must pervade each and every ESSF activity. Avoid the feature trap by establishing a goal of improving attack resistance in each activity from its inception. Some guides:

- construct misuse/abuse cases,
- model the threats each application faces,
- assess applications against a threat model, misuse/abuse cases,
- train using vulnerability case studies,
- define standards based on risk, vul-

nerabilities,

- avoid relying on security feature checklists, and
- avoid relying solely on API-guide security standards and training.

Like adopting framework activities, attempting to adhere to each of these guides on day one can be too onerous. Build on your organization’s current strengths, infusing this guidance opportunistically.

Possess five competencies

Regardless of how an organization operates, every good ESSF addresses five pursuits in one form or another. Organizations should iteratively raise their competencies in each of these pursuits gradually as they adopt their ESSF.

Enterprise software security framework

The ESSF defines an organization’s approach to software security and describes roles, responsibilities, activities, deliverables, and measurement criteria. It also includes a communication plan for en-

terprise-wide roll out.

Enterprise software and data architectures are essential anchors of the goal-state an ESSF defines. Definition of and migration toward a secure enterprise architecture is thus part of the framework competency.

Knowledge management, training. An organized collection of security knowledge is likely to include policy, standards, design and attack patterns, threat models, code samples, and eventually and eventually a reference architecture and secure development framework.

Another element of this competency is the development and delivery of a training curriculum. Topics include security knowledge as well as help for conducting assurance activities. This pursuit also includes new courseware, along with retrofitting of existing courseware to software security concepts.

Security touchpoints. The definition of tasks and activities that augment existing development processes (formally or informally) help developers build security into any custom software development process, as well as in-place outsource assurance and commercial-off-the-shelf validation processes. This competency defines how to assure software.

Assurance. The execution of security touchpoint activities provides assurance—conducting a software architectural risk assessment, for example, validates that security requirements were translated into aspects of the software’s design and that the design resists attack. Assurance activities rely heavily on the knowledge and training competency to define what to look for.

Tool adoption is likely to be part of this pursuit in the short-to-medium term. It will involve the purchase, customization, and roll out of static analysis tools as well as dynamic analysis aides. Your organization might have already adopted a pen-

etration-testing product, for instance.

Governance. In the context of an ESSF, governance is competency in measuring software-induced risk and supporting an objective decision-making process for remediation and software release. This competency involves creating a seat at the project management table for software risk alongside budget and scheduling concerns.

Governance should also be applied to the roll out and maturation of an organization’s ESSF. The framework’s owners can measure project coverage and depth of assurance activities, reported risks (and their severity), and the progress of software security knowledge and skill creation, among other things.

Define a Roadmap

Each competency depends somewhat on the others, and growing each effectively demands thoughtful collaboration. It’s foolish to attempt to understand all the subtle interdependencies from the start and attempt a “big bang” roll out. Instead, good ESSFs leverage key initial successes in support of iterative adoption and eventual maturation. Keep two things in mind:

- *Patience.* It will take at least three to five years to create a working, evolving software security machine. Initial organization-wide successes can be shown within a year. Use that time to obtain more buy-in and a bigger budget, and target getting each pursuit into the toddler stage within the three-year timeframe.
- *Customers.* The customers are the software groups that support the organization’s lines of business. Each milestone in the roadmap should represent a value provided to the development organization, not another hurdle.

Thankfully, the organizations that have been doing this work for a few

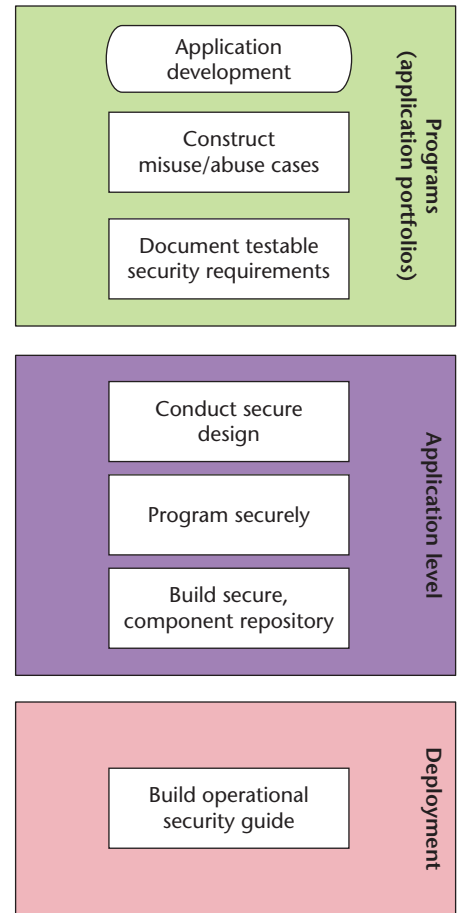


Figure 2. Role activities: what. In this relative ordering, teams know which activities depend on others. Providing any more of a detailed “when” diagram can be seen as offensive and overly constraining to each suborganization. Let people conduct detailed project planning around these activities themselves.

years now are starting to share some of their experiences. Expert help is increasingly available, too. As always, use your community resources, and good luck being the agent of change in your organization! □

John Steven is a technical director and software security principal at Cigital. His interests include J2EE security, and he works in partnership with companies large and small to help them build their own software security capabilities internally. Steven has an MS in computer science and a BS in computer engineering from CASE, in Cleveland Ohio. Contact him at jsteven@cigital.com.