10 Quick, Dirty, and Cheap Things to Improve Enterprise Security

James McGovern
The Hartford

Gunnar Peterson
Arctec Group

As software security has increasingly become an important part of information security programs, there have been some notable trends and successes of various tools, processes, and models. Because "building security in" is so different from how enterprise software has historically been developed, the changes might seem revolutionary. In the enterprise, revolutionary changes involve cost and complexity, as organizations figure out how to incorporate new techniques, processes, and technology.

However, building (some) security into your enterprise software doesn't require that you don revolutionary berets and grow beards like Fidel and Che; evolutionary progress doesn't always require cross-functional teams, a project management office, and other enterprise foofaraw. What follows is an informal list that doesn't say, "simply reboot your entire enterprise software development and you are ready to begin secure coding." Instead, we describe how people with limited budgets and/or authority can make potentially big changes in their enterprise software's overall security. Without further ado, here are 10 low- or no-cost ideas.

1. Don't Just Market to Executives, Market to Developers

Secure software development aims to design, implement, configure, and sustain software systems in which security is a necessary property from ideation to production and therefore has many players. Too often, the marketing of application security is targeted at executives. Although it's important to gain business buy-in, it's equally important to ensure developer buy-in. After all, the BlackHat community isn't really interested in exploiting the work product of project managers or business analysts. They have a laser focus on breaking software, the work product of developers.

At first, marketing to developers will generally be difficult because it will uncover faults in their work product that the quality assurance (QA) department previously missed. Some developers might initially show resentment, especially if the marketing unfairly compares one developer with another. However, most IT developers want to write better code; if you give them tools that help find defects in design and at code review time, you can easily obtain buy-in.

If all else fails, a proven approach to reach skilled developers is to appeal to their "inner engineer"—that is, pull on the heartstrings of engineering's quest for reliability. No one starts a career as a software developer with the goal of having his or her software hacked to pieces. A little reminder of why we're all here can go a long way.

2. Leverage Container Services Wherever Possible

Early Java application servers didn't provide security functionality to developers, so they had to craft their own. Today, modern application servers, whether Java, .NET, or Ruby, all provide security services that developers can use to build a better security model for their applications.

Many containers provide reusable components or services that aid in security, including functionality around authentication, authorization, role mapping, logging, and security externalization or integration where appropriate. Many containers even support standards such as the Security Assertions Markup Language (SAML), with which you can potentially gain single sign-on (SSO) without writing additional code, provided you didn't reinvent the wheel. The app server container is bought and paid for; why not leverage its security features?

3. Encourage a Community Orientation

It's been said (ironically) that most organizations can't afford to do it right but can afford to do it twice. If you can't afford to train your developers, you can at least encourage them to attend local user groups. Organizations such as the Open Web Application Security Project (OWASP) have organized user groups in more than 130 locations throughout the planet. These groups meet after hours to discuss a variety of application security topics, and the meetings are free.

On the remote chance that none of the locations are in your area, online forums are the next best avenue. Online communities such as LinkedIn and Stack Overflow let you gain perspective on topics ranging from how to find cross-site scripting vulnerabilities in Smalltalk all the way to how an enterprise architect can sell a security strategy to his or her organization's board of directors. You don't need an expensive global consultancy or even analyst firms to help you craft a strategy; you might find that many in the community will share their PowerPoint presentation with you for free.

4. Focus Less on the Process and More on the Competence

Although there's often only one right way to do something, the number of ways to do things insecurely is infinite. You can't checklist your way through security. So, although many enterprises focus on processes and suffer from analysis paralysis when comparing security process frameworks such as the Comprehensive Lightweight Application Security Process (Clasp) or Touchpoints, there are likely no barriers to building a threat-modeling center of excellence. Best of all, you already own the best threat-modeling software—it's between your ears.

Threat modeling follows the threat's vector through known vulnerabilities. You use the vector to identify and locate countermeasures in the system's software, data, and identity layers. Threat modeling provides several benefits, including these:

- It lets you provide a business justification, by mapping threats to business assets, for security features at all layers of the stack for identified threats.
- It enables a thoughtful conversation around risk and trade-offs in an objective, quantifiable way.
- It encourages a logical thought process in defining an application's security model.
- It lets architects and developers work together to find threats at design time and build security in, instead of hoping that QA can discover those threats later in the life cycle.
- It helps business analysts understand and create nonfunctional traceable security requirements.

In security, having repeatable results is more important than having repeatable processes.

5. Minimize the Ways to Attack an Application

In many company cultures, "change management" is a code word for preventing change. This causes developers to take steps to make their lives easier by incorporating back doors into applications, not cleaning up dormant code, or even figuring out clever logic to bypass authorization to monitor the well-being of applications in production. The challenge is that dormant code and back doors could also be viable attack vectors. Change management must become a developer's best friend, not his or her worst nightmare.

Even when developers aren't taking any of those steps, some are still guilty of other practices such as developing software using a local administrator's account, which might not expose unnecessary application privileges until someone exploits them. We should encourage developers to use administrator privileges only when they're administering. Otherwise, when developing software, they should have even lower privileges than a regular user. This is best done by giving developers two machines but can be accomplished by giving them two credentials: one for highly privileged access and one for general-purpose use.

6. Prioritize Your Security Needs

You can prioritize security remediation in many ways. The default should be a risk-based approach using a framework. (For ideas, visit OWASP's risk-rating site; www.owasp.org/index.php/OWASP_Risk_Rating_Methodology.) Other times, you might need to do something simply for compliance reasons or even because a particular security request might be a marketable feature in management's eyes. At the end of the day, it's best to have thoughtful governance around how to prioritize risk, compliance, and feature orientation, so that you never send a mixed message and the developers don't work on solving the wrong problem. When spending your precious time and money on security, make sure the prioritized issues get the attention.

7. Find Diamonds in Your Backyard

This might shock some battle-weary, jaded enterprise security people, but good things are probably happening right now in your organization with regard to security. Even amid the myriad challenges enterprises face in developing more secure software, some existing developer behavior will likely meet or exceed security requirements, whether it deals with input validation classes, instance level authorization, audit logging, or secure exception-handling classes. Your job is to find working examples that are proven in your enterprise and "bless them" as good examples of security standards, patterns, and practices.

With the only cost being your time, you can help get wider adoption of secure software that's proven to work in your enterprise while building goodwill with developers by recognizing good work, not just finding faults. This approach has the convenient byproduct that for the existing good practices, you don't spend your precious time and political capital lecturing birds on flying.

8. Improve Your Audit Logging

In information security, there aren't many areas in which the defender has the advantage. But there is usually at least one: knowing your assets. Securing assets isn't simply a matter of adding protection mechanisms such as authentication and authorization; it's also a matter of detection and response through mechanisms such as audit logging. Beyond security, detection and response mechanisms have economic advantages over protection mechanisms. For example, introducing a new access control scheme (say, swapping a username or password for Kerberos) will involve a major development, QA, or deployment project. This might be beneficial, but it won't be cheap. A less expensive alternative is to add a passive logging system that need only be aware of certain asset-centric events and write them to a log management system for audit review.

Security information and event management (SIEM) tools can add visibility to security events and incidents in your enterprise. However, the real value comes with layering the audit-logging system to report on the assets you know and value most. The open source Distributed Audit Services (XDAS; www.opengroup.org/projects/security/xdas) project provides a useful starting point for the audit-logging model's two critical elements: the event model (what events are audited) and the audit record format (what's in the payload).

9. Send in the Crash Test Dummies

There's nothing like random data and behaviors to help you learn how your application will deal with random data and behaviors. Fuzz testing—using such input to test your software—is invaluable for finding bugs while your software is under development and in QA. Vulnerability assessments from "clueful" firms can be very valuable—and expensive. Several good open source or free fuzzers are available; mastering one helps you understand how the application will fail and gives you

- an up-close, personal view of the application, data, and users;
- a health check of the application; and
- repeatable test cases to see whether the fixes work.

Then, when you do engage the clueful firm for vulnerability assessment, it can work on second- or third-order problems.

10. RTFM (Read the Flippin' Manual)

Most enterprise software (such as IBM Websphere, Oracle, and .NET) has detailed guidance on how to lock down your deployments. Have you read and parsed these guidelines for your high-value assets and created a security deployment checklist for your application, database, Web servers, and so on? The security baseline is a way to align efforts with system administrators who are often happy to aid security efforts but might lack the detailed knowledge and understanding of software to enforce specific configurations—for example, the Lightweight Directory Access Protocol (LDAP) over Secure Sockets Layer (SSL) rather than LDAP in the clear. The RTFM recommendation goes double in enterprises that are more buy than build, in which systems are more configured than coded.

11. (Bonus Tip) Let Developers Be Successful

Organizations have many practices that are generally sound but might need adjustment to incorporate a sustainable security culture. Sometimes, developers aren't allowed to make changes unless the QA team finds a bug or the change is a feature request prioritized by management. Although it's not a good practice to give developers total freedom, it's vital that if they see something that can be improved, you give them the opportunity. Professionals take pride in their work product and will appreciate the opportunity and management's support for delivering higher-quality software.

Another challenge in many corporate cultures, especially when IT leadership might be less technical, is perception management. When security professionals find vulnerabilities, it can feel like they're always raining on someone else's parade, which is simply the wrong attitude. Security needs a healthy dose of confrontation, and separating the message from the messenger is vital. If such separation isn't possible, you should at least instill a sense of amnesty. Nothing will destroy a thoughtful security program faster than a developer who wants to do the right thing for the organization getting beat up by that organization for the wrong reason. □

James McGovern *is an enterprise architect for the Hartford and a technology advisor to business enthusiasts. Contact him at james.mcgovern@thehartford.com.*

Gunnar Peterson *is managing principal of Arctec Group. Contact him at gunnar@arctecgroup.net.*