

If Security Auditors are from Mars and Developers are from Venus...

By John Viega

john.viega@securesoftware.com

Security and development teams must learn to speak a common language to integrate security into the application development lifecycle and improve overall software quality.

ChoicePoint, CardSystems, LexusNexus, Polo Ralph Lauren. As headlines continued to be littered by cases of high-profile security breaches, businesses are being held increasingly accountable for the security of their applications by customers, partners and government. Poor application security can result in heavy downstream remediation and management costs, as well as productivity problems, hits on revenue, compliance issues and damage to corporate reputations.

Unfortunately, most organizations are so busy playing catch-up with security that they neglect the application security problem. They have invested in network-perimeter protection, application-security gateways, and manual software audits. But these approaches are largely after-the-fact solutions that do not target the root cause of security: security flaws in the underlying software.

The Weak Link in the Security Chain

Application security is an enormous, poorly addressed vector of risk for many of the world's largest organizations. For example, overt problems in software—such as SQL injection session hijacking and buffer overflows—are caused by extremely common coding mistakes. Although they are easily corrected, these security defects often go unchecked during the software development lifecycle. As a result, these vulnerabilities provide an avenue for many of the most common attack types against corporations—attacks that result in millions of dollars worth of productivity losses, data theft and the like.

So how big is the application security problem? Gartner estimates that approximately 70 percent of all attacks happen at the application layer, and that it is vastly less expensive for all involved—including the development organization and the customer—to remediate vulnerabilities during development rather than post-deployment. Most security breaches that lead to identity theft, network outages, data loss, or Web site defacement have a root cause in a security flaw that was the result of poorly written software code.

As a result, application security is an important emerging requirement in software development. Beyond the potential for severe brand damage, potential financial loss and privacy issues, risk-aware customers such as financial institutions and governmental organizations are looking for ways to assess the security posture of the products they build or purchase. These types of organizations ultimately plan to hold vendors accountable for security problems in their software.

It is clear that security administrators and development teams agree that the best long-term answer to the security problem is to fix these common

problems and to make software intrinsically more secure. Unfortunately, this is much more easily said than done.

Who Likes Change?

Addressing application security problems effectively is difficult, because the traditional software development lifecycle does not deal well with these concerns. This is because software developers lack structured guidance—the few books on the topic are relatively new, and they only document collections of best practices.

Also, development organizations generally prefer to focus on core functionality features, addressing security in an ad hoc manner during development. But given their limited security experience, developers typically provide a minimal set of services. This usually results in over-reliance on poorly understood security technologies.

For instance, secure sockets layer (SSL) is the most popular way to provide data confidentiality and integrity services for data traversing a network. However, most SSL deployments are susceptible to network-based attacks because the technology is widely misunderstood. Particularly, people tend to treat SSL as a drop-in for traditional sockets, but when used that way, critical server authentication steps are skipped. Performing proper authentication is usually a highly complex process. Organizations that deploy technologies such as SSL and Java are often susceptible to a false sense of security.

To add to the problem, while most security professionals recognize some of the common pitfalls, they are unable to communicate clearly with developers and cannot implement changes to the development lifecycle. Unfortunately, most organizations do not even see the language gap between developers and traditional security professionals. They don't realize that asking developers to add security to a product already in development is akin to asking an auto manufacturer to install seat belts, airbags and a steel-enforced, roll-over proof cabin into a car after it has hit the assembly line. This ignores the fact that software development is a process, and that the only way to impact the quality of the end product is through changes in the development process.

The result of the typical shoestring approach to software security is the so-called "penetrate and patch" model. Organizations cross their proverbial fingers, hoping that security problems will not manifest themselves and deferring most security issues to when they do appear—which is often well after software deployment.

Of course, bolting on a security solution when a problem is found is just as nonsensical as adding on a reliability module to fix robustness problems after software is developed. Again, industry research has examined the costs of addressing security issues at various points during the develop-

ment lifecycle and clearly shows that the cost of deferring security issues from design all the way into deployment is as much as 10 times greater than the cost associated with traditional reliability bugs. This is largely due to the tremendous overhead that accompanies vulnerability disclosure and actual security breaches.

Security professionals want to help developers write better code, but have always lived in a world where the generally accepted solution is to simply throw more software at a new security problem. The most recent example of this reactionary nature of the security market is the proliferation of spyware and the resulting emergence of the anti-spyware market. Organizations have become accustomed to finding medicine to treat the symptoms of the security disease rather than trying to cure it altogether.

In what may be the most telling sign that the industry's efforts are misguided, recently-published research from the SANS Institute shows that hackers and virus writers are now aiming at the actual security products that corporations use to protect themselves. Anti-virus applications are among the most targeted pieces of software by hackers now that operating systems seem to have stopped some of the bleeding.

So hackers are now attacking the software that protected our software. Does that mean we are supposed to add yet another layer? Are we going to deploy new software to protect the software that was protecting our software? Confused? Don't worry, you are not alone.

Identifying the Problem

Organizations need to realize that development professionals live in a world vastly different than security professionals. Development is a process-driven discipline where steps and roles are extremely well-defined, and upsetting the process can result in product development and shipment delays—an outcome that can make management, sales and even shareholders very unhappy. Development organizations are driven by time-to-market and new feature pressures, not by the need to write more secure code. Only in the most high-profile cases do security breaches result in some sort of action taken by the development organization to rectify the situation during development.

Security has reached the mainstream conscientiousness of all areas of business and society. Compliance requirements—including Sarbanes-Oxley and other regulations—have moved security to the top of the agenda in the board room of every large company. High-profile thefts of personal information like those at ChoicePoint and CardSystems have put security back into the media spotlight as a top concern for consumers. Isn't it time that security moved into the one area where it can make the biggest difference for all vendors of software, builders of applications, and the people who use them?

Security administrators and developers need to work together to push security into the early stages of the software development lifecycle in order to address the root cause of the overall security epidemic. In order to do so, organizations are going to have to build consensus among security, development and management that better application security is a priority. Unfortunately, most don't know where to start.

Bringing Development and Security Together


There are several steps that organizations can take to get started down the path to better application security. These include:

1. **Institute awareness programs:** Educate the organization on what

is important, why and who is accountable.

2. **Establish assessment strategy:** Determine what the inspection process will be and how the results are to be analyzed.
3. **Establish security requirements:** Ensure that security requirements have the same level of "citizenship" as all other "must haves."
4. **Define and monitor metrics:** If it is not measurable, progress is impossible to determine.
5. **Implement secure development practices:** Defined security activities, artifacts, guidelines and continuous reinforcement must become part of the culture.
6. **Build vulnerability remediation processes:** If it is bad and you find it, you must be able to assess and contain the exploitation potential, and collapse the problem.
7. **Publish operational guidelines:** The safe-handling procedures for the security of an operational system; if a problem is discovered and the system cannot be fixed immediately, the team must be informed of its options.

These basic steps are not the cure-all to the application security woes, but they are a good start. Working to rid applications of vulnerabilities involves an agreed upon lingua franca within the software industry. The industry needs to develop a standard for integrating security processes, roles and artifacts into the existing application development lifecycle with minimal pain and impact on time to market.

Even if the approach is modular and takes time to implement, a communications standard will still mark an all-important first step to improving application security. Such a standard is a must if we are going to truly change the way that software is developed and impact the overall quality of software. 

John Viega, CTO and co-founder of Secure Software, is author of Building Secure Software: How to Avoid Security Problems the Right Way (Addison-Wesley Professional, 2001).