



# The Seven Deadly Myths of Software Security

---

## Busting the Myths

With the reality of software security vulnerabilities coming into sharp focus over the past few years, businesses are wrestling with the additional risk that poor security introduces. And while the risk is becoming clearer, methods to defend applications from attack remain murky. Further clouding the picture, the responsibility for application security tends to fall organizationally in a netherworld between the offices of the CSO (compliance and risk), the CTO (application development), and the CIO (information operations). All three groups are committed to the business succeeding (which also means keeping the business safe), but their charters and approaches tend to be very different. For any given aspect of security or functional role within the organization, one can find lists of “best practices” from a wide range of sources. While these lists can sometimes be helpful, unfortunately, they have led to many “myths” about application security that have taken root. In this paper, we outline some of the most prevalent myths about security that you should consider when looking to improve the security of your software. Falling prey to these deadly myths could at best cause you to waste valuable cycles on useless “security” activities or at worst, cause your applications to be *less* secure.

---

## Myth #1: Compliance = (Application) Security

**Myth 1.1:** *Legislative requirements and standards are well aligned with business risk and security because the individuals (or committees) that wrote standard/requirement/law X understand real business security risk and real software security risk.*

**Myth 1.2:** *If your company passes a PCI/SOX/SAS 112/... audit, then it is equivalent to passing a serious security evaluation.*

**Reality:** Regulatory compliance and security are rarely well aligned, especially when it comes to applications. Many well-known (and well-intentioned) standards either ignore application security or address it superficially (by only touching on web applications for example). Some standards were written to remedy very specific problems. Sarbanes Oxley, for example was about accountability for reported financial information, so being in “compliance” with it says little about application security risk. In addition to narrowness of purpose, some compliance standards focus almost exclusively on network security. For example, the Payment Card Industry Data Security Standard (PCI DSS) has some specific requirements for network security, but currently only superficially addresses application security. The industry agreed-upon need to integrate security throughout the entire software development lifecycle is not addressed by these well-intentioned standards.

**Myth 1.3:** *External/internal auditors understand software security and would catch any big problems.*

**Reality:** Most compliance auditors do not have a software development background. This, combined with standards that don’t touch on the specifics of software security means that even big software security problems are easy to miss. A company that is “compliant” with standards can therefore be at high risk from vulnerabilities at the application layer.

## Myth #2: “We don't have a software security problem.”

**Myth 2.1:** *We haven’t had any major security breaches so we don’t have to worry about security.*

**Reality:** Lack of a known security breach is not a good indicator of security. In the absence of a major failure, and given the fact that many regulations are nebulous about software security, one might be inclined to believe that software security is not a problem. In reality, an application vulnerability could create the potential for a type of breach that most IT departments are just not equipped to detect. Also consider that if you haven’t been attacked, it doesn’t mean that you won’t be. Automation and new techniques now make it cost effective for attackers to go after a widening range of targets. Finally, even extensive quality-focused testing is likely to miss some big security problems, therefore a clean bill of health from QA is not necessarily a good indicator of security.

***Myth 2.2: We don't have many web applications, so security isn't a problem.***

***Reality:*** Web applications do pose a significant risk, but vulnerabilities in non-web applications should be taken just as seriously. Imagine a flaw in an internal (non-web-facing) server application that processes customer data. Ignoring the security of that application is tantamount to trusting every piece of data it processes. Unfortunately, in most cases, you cannot trust every piece of data that your system processes. Malicious data may simply “pass through” an entity that is “trusted” (like a database) and reach vulnerable code in ways that you never expected. In a world of constrained resources, one might be willing to take that gamble with applications that have a low consequence of failure or are only touched by carefully sanitized data. However, for applications that are critical to the business or process data of value, this risk can be severe and the consequences costly.

***Myth 2.3: We don't fall under SOX/GLBA/HIPPA/... so security isn't a major issue for us.***

***Reality:*** Software vulnerabilities can put key data and business processes at risk, an issue that goes well beyond compliance. Security must be considered for any application where confidentiality, integrity, privacy, or availability is important.

## **Myth #3: Network defenses will protect us**

***Myth 3.1: Software security vulnerabilities are neutralized by network defenses (such as routers and application firewalls) so we can defend against most attacks at the network level.***

***Reality:*** Many network security solutions make bold security claims, with some reaching the level of panacea. The reality, however, is that many network security controls *assume* that software is secure instead of actually protecting the enterprise against software security failures. For example, if properly used, SSL can create a private tunnel between a user and a server application. It does little to protect the business however if the user is malicious and the application processing his or her data is vulnerable. Even good application firewalls that can correctly identify many straightforward SQL Injection or Cross Site Scripting attacks cannot defend against business-logic security vulnerabilities or buffer overflows that might reside in software that is processing user input. If we look at the application lifecycle, a firewall should be one of the last steps in a comprehensive security program, not the only step and not the first.

## **Myth #4: Software security is someone else's problem**

***Myth 4.1: We're already handling it. The CSO may think: "The software development group is dealing with application security since they are building the applications. I just need to worry about other aspects of security." The CTO/CIO may think: "The security group is dealing with security of our applications since they are the experts."***

**Reality:** There are many phenomenal CSOs, CTOs, and CIOs in the industry: individuals that leverage their budget and team appropriately to both enable and defend the business. Software security rarely, however, fits neatly into any of these groups' specific charter. It is also rare that these individuals have both a strong security and software background. Being a CSO requires a diverse skill set: diplomacy, paranoia, marketing, advocacy, execution and management. Their backgrounds vary wildly: technology, law enforcement, management, academia, etc. Most have a very good grasp of security concepts but have not written software and are not familiar with the risks that a vulnerable piece of code can bring to an enterprise. Similarly, those who rise to the office of CIO or CTO have rarely seen the trenches of security. The result is a disconnect and the belief that software security is a natural outcome of either a good software development group or a good network security group. The reality is that both are required. Software security demands specific and focused effort through collaboration between the groups responsible for defending the business and the groups that build the software to power it.

**Myth 4.2: Attackers go after platforms and widely used applications; they won't look for issues in our proprietary software.**

**Reality:** A maturing underground economy has now made it easy to turn stolen data into cash. As a result, smart people are willing to spend significant time, effort, and money to attack specific targets. Combine this with the widespread availability of tools to "test" network-facing application interfaces at a distance and you've got a clear business case for attackers looking for vulnerabilities in proprietary software.

## **Myth #5: Magic bullet theory**

**Myth 5.1: A penetration test finds all security vulnerabilities.**

**Myth 5.2: A source code scanner will find all security vulnerabilities that the business should care about.**

**Myth 5.3: Java, C#, etc. are secure languages. Using them eliminates most (or all) security issues.**

**Myth 5.4: I do X for security and therefore it's covered.**

**Reality:** For any problem, it is natural to seek out (and hope for) one single all-encompassing solution. It might be a tool you can buy, a service you can pay for, a person you can hire, etc. Software security is not such a problem. For example, consider these popular software security solutions:

- Penetration testing: This can be helpful, but it will not find all the major vulnerabilities in an application. It can also be very expensive to fix vulnerabilities if they are found late in the software development lifecycle.
- Source code scanning: An important part of an overall solution, but should be coupled with processes that help identify problems in design and deployment.

- Application firewalls: Tend to cover a narrow set of threats; no substitute for integrating security into the software development lifecycle.
- “Safe” languages like Java, C#, etc. – Help mitigate some vulnerabilities (like buffer overflows) but do not defend against other important vulnerability types.

Security needs to be implemented throughout the software development lifecycle to be effective. One must gather the real security requirements for a system and consider compliance, safety issues, contractual requirements, what data the application will process, and business risk. During design, it is important to ensure the architecture is secure by building threat models and applying principles like *least privilege* and *defense in depth*. During development, security training is key and should be coupled with the use of source code scanners. Testing must consider the attacks generated by tools but must also include the creativity of a human. During deployment the application must be configured to work securely in its environment. No point solution covers all of these needs. Security tools and processes like education, threat assessment, and source code scanning are necessary to address the problem but none of them in isolation is sufficient.

## **Myth #6: Hopelessness: Security is too costly so I’ll take my chances**

***Myth 6.1: Software security can’t be measured therefore it’s hard to make a business case for it.***

***Myth 6.2: The overhead of software security is going to be crippling to my development team and therefore it is too expensive..***

***Reality:*** Along the road to software security improvement, some have detoured to hopelessness –the belief that a lack of business metrics, the scale of the problem, or the demand on development teams are too great to make progress. The reality is that software security is not quick or easy but there are incremental improvements that can be made to significantly reduce risk. Several software security oriented efforts have emerged such as:

- Software Assurance Forum for Excellence in Code (SAFECode) – a consortium of leading software vendors focused on practices to improve software security.
- Microsoft’s Security Development Lifecycle (SDL) – Microsoft has released a detailed set of practices (and some free tools) that help to improve software security.
- The U.S. Government has helped to better frame the application security problem through identifying and monitoring vulnerabilities with its Common Vulnerabilities and Exposures (CVE) and Common Weaknesses Enumeration (CWE) efforts.
- The Department of Homeland Security (DHS) has built a portal called “Build Security In” (<http://buildsecurityin.us-cert.gov>) to gather secure development practices and also sponsors collaborative efforts with industry such as Coverity’s open source scanning site (<http://scan.coverity.com>).

These efforts, along with a willingness for some enterprises to talk about their application security experiences, has painted a much clearer picture of application security processes that work and those

that do not. Almost all of the widely used methodologies agree on a few key elements: security education, threat modeling, source code scanning, and security-oriented testing. Making incremental progress in these areas has a synergistic impact on application security. There are also indications that key standards (such as PCI DSS) will focus more on software security as they evolve. This will create a tighter measurement tie-back of software security to compliance and risk management. As a result, there is reason for hope: software development teams can absolutely implement effective security processes and products while protecting budgets.

## **Myth #7: Outsourced code is completely vulnerable [or completely secure]**

***Myth 7.1: Outsourcers build secure software.***

***Myth 7.2: It is impossible to build “security quality” into outsourced development contracts.***

***Reality:*** Many enterprises have opted to move development of some products to outsourcers, many of them off-shore. In addition to the management challenges of outsourced software development, businesses give up a degree of control over who is developing their code, the background of the individuals, and the policies and processes they adhere to. The transactional nature of outsourced development has led to the emergence of two *completely opposite* myths: either the software produced by outsourcers is of good security quality, or alternatively, the software produced by outsourcers is of inescapably poor security quality. In practice most outsourcers are beholden to contractual customer acceptance tests for software. If *security* acceptance tests and processes are not a part of a development contract, it is unlikely that the result is software that an enterprise can deploy with confidence. Software security requires focused effort. It is not a natural outcome of conventional software development processes, even from development groups that have good traditional “quality” practices. Therefore, even outsourcers that have highly mature and repeatable processes, as indicated by a high Capability Maturity Model (CMM) rating for example, do not necessarily produce more secure software. One option is to bake security benchmarks into development contracts. Possibilities include enforcing basic software security training, requiring copies of design and threat modeling artifacts, and specifying satisfactory results from tools like source code scanners and penetration testing tools. Some companies have managed to do this very successfully; in addition to reducing risk for the externally developed software, it can also indicate an added degree of security diligence to auditors.

## Summary

Unchallenged by metrics and data, several dangerous myths have managed to spread widely. Thanks to breach notification laws (which have given us some data), new processes, better tools, core research and security education, reason may be making headway against myth. The security landscape has been quickly reshaped in the past few years. Systems are interconnected. The “network perimeter” may no longer exist. Attackers have become organized and profit-driven. In this environment, it is vitally important that businesses reexamine some of the “conventional wisdom” around software security. As an industry, we are still an impossible distance away from building unbreakable software, but we have learned a lot. We know that software vulnerabilities can represent a severe and pressing threat to businesses. We know that security is not a natural byproduct of the way we have been building software. Most importantly, we know that making security improvements to development processes is possible, practical, and essential.

Copyright © 2009 People Security Consulting Services LLC. All rights reserved.

People Security is an application security training and consulting company. Visit us at [www.peoplesecurity.com](http://www.peoplesecurity.com).

THE INFORMATION IN THIS PUBLICATION IS PROVIDED “AS IS.” PEOPLE SECURITY CONSULTING SERVICES LLC MAKE NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS REPORT, AND SPECIFICALLY DISCLAIM IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR USE.