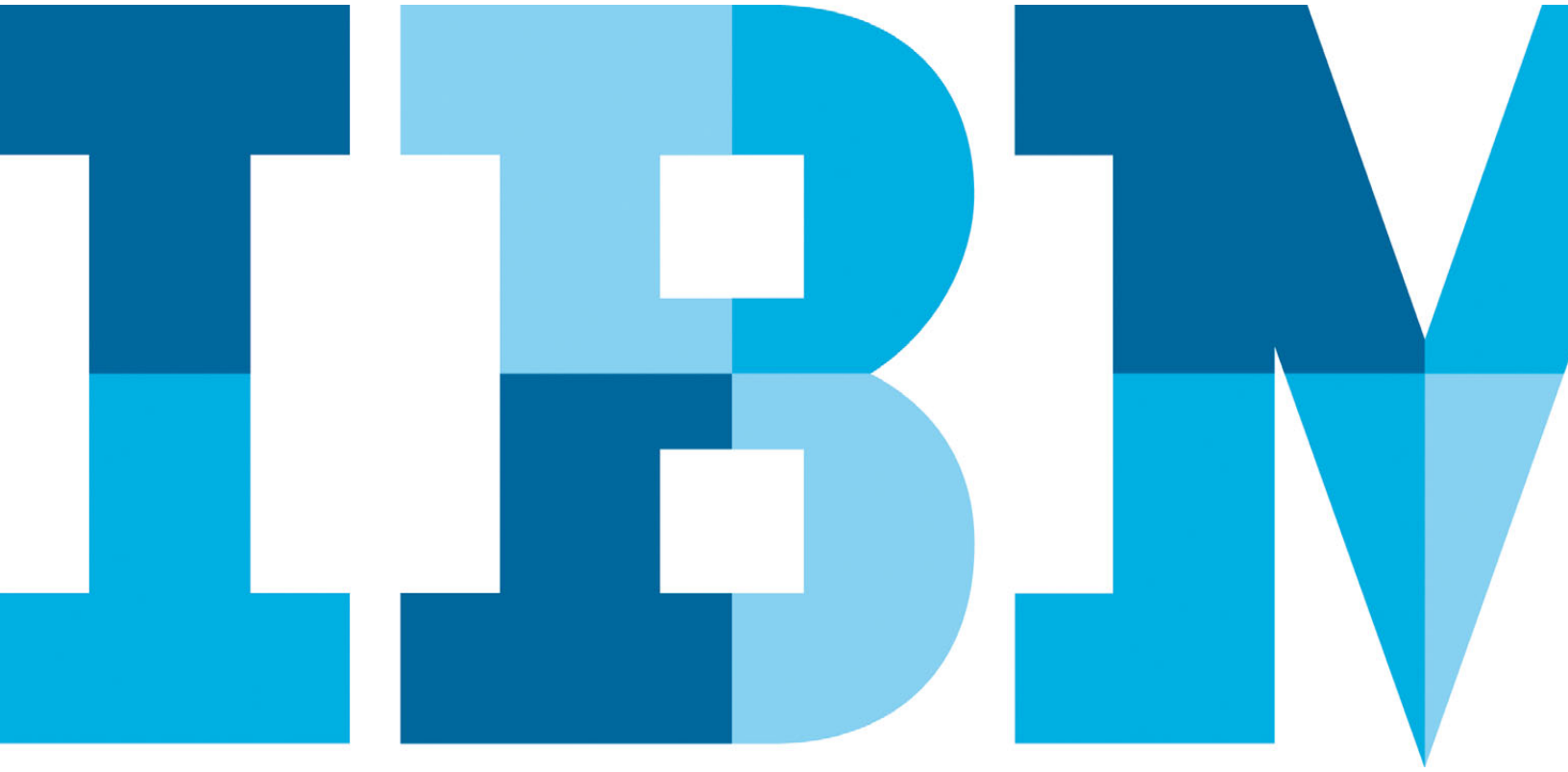
Motivations for software security: An executive overview



IBM

Contents

Software has driven corporate value like no other business capability for the past decade. As the planet continues to become more technically instrumented and interconnected, software will allow businesses of all types to derive intelligence from innumerable data sources in new, innovative and smarter ways.

Software is created in many different ways: commercially available packages, internally developed and proprietary components, applications that are outsourced or off-shored, systems assembled from multiple modules, and, most frequently, a combination of these models. One trend that has been consistent is the tendency towards web-based applications and solutions delivered via web services. This trend has been spearheaded by the business possibility of global markets, mobile employees with multiple access devices, and the challenge of keeping thick-client software fully patched and synchronized in large multiuser environments. Web applications are attractive to the business because they allow anyplace-anytime access to critical business services, for both internal and external users.

In parallel with this increased dependence and natural attraction to web-enabled applications comes a dash of cold reality: software exposes the business to risk. Over the past decade we have seen the evolution of security vulnerabilities shifting from the network and infrastructure layer to the application layer—which means a vast proliferation of vulnerabilities: today nearly 50 percent of all security vulnerabilities reported through various channels are being found in web applications.[1] Not only has security research proved that software is vulnerable, but reports such as the IBM® X-Force® Security Trends Report demonstrate that attacks are becoming more prevalent as well. Sifting through the seven billion security events that IBM witnesses on a daily basis,[2] specific web-based attacks such as SQL Injection (an injection vulnerability where a malicious payload is passed on to a database without the proper validation or sanitation) appeared more than one million times each day in 2009.[3] Not only are these dangers present, but statistics suggest that organized criminal elements are using automation to discover and exploit a generally vulnerable environment. When you consider the aggregate threat—the likely number of security vulnerabilities, the prevalence of attacks, and the average organizational cost per data breach at $6.75 million[4]—it's clear that we have a business challenge that demands attention.

Understanding the unique motivations for introducing security improvement will lead to better business decisions that allow the business to thrive in a hostile operating environment. This paper describes four motivating factors for improved software security, and it concludes with an overview of the evolution of software vulnerability that can help you understand your own security challenges.

Improving software security

How should businesses address security challenges? A single solution cannot apply to everyone. It may involve external security testing, establishing internal processes for security assessment and remediation, or a culture of security applied to software design, development, and delivery as outlined in the IBM Secure Engineering Framework.[5] All of these approaches yield a financial and resource impact on the business.

We are faced with some difficult decisions when it comes to software security. Given the dependency of business on cyber initiatives, typical questions include: how much should I invest in security improvement? What is the right kind of investment? How can I combine a security initiative with the need for code compliance set by industry standards and government mandates? Many of these answers will depend on the business motivation for improving software security.

Return on investment

While return on investment (ROI) as a software assurance motivation has been hotly debated with passionate arguments throughout the software industry, it seems rational to measure the return to be achieved with each dollar invested.

Those in support of a security ROI calculation argue that planning for and mitigating security issues early will, at minimum, result in cost avoidance.[6] Take an example from the auto industry. It is much cheaper to find and correct an engineering issue early in the design phase than to invoke an automobile recall and incur significant costs in post production. Studies have shown that finding an issue post production can cost as much as 100 times more than finding the same issue early in the development process.[7] In the worst case scenario, where the vulnerability leads to a software breach, the ROI calculation must also include revenue losses for fines, negative brand impact, and loss of customers.

Those in opposition to security ROI analysis argue that security incidents are singular events, and that security investment is more of an insurance policy. It is a cost to the business. Having vulnerabilities in the software code cost nothing until regulatory requirements (with potential fines) dictate otherwise or until the vulnerability is actually exploited. "Running the ROI calculations," they say, "stipulates in advance that there is a certainty of business impact and no one can know in advance what the business impact will be."

This conundrum has impacted the security industry at large. Investment in improved security is often only addressed when the pain is sufficient or when an executive mandate is present. Security solutions tend to be very reactive in nature. True security innovation is surprisingly rare given the size of the market.

Regardless which position you take in the security ROI debate, determining the ROI model for investing in secure software requires calculations that are unlike those used for most other areas of business operations.

Compliance

Many businesses do not have the option of choosing whether or not to invest in secure software. Regulatory requirements mandate an executive focus on application security. If you are operating in the healthcare environment, you have faced the Health Insurance Portability and Accountability Act (HIPAA) of 1996. If you are operating in the financial world, you are no doubt aware of the Gramm-Leach-Bliley Act (GLBA) of 1999. And if you are collecting or working with credit card information, the Payment Card Industry Data Security Standard (PCI DSS) is a regulatory requirement that mandates an Application Security Program.

Using regulatory compliance as a motivation for software security, however, can be a double edged sword, since it can both benefit and challenge the organization.

Advocates of regulatory compliance correctly point out that an industry bar has been set that ensures a minimum level of software security. Compliance is beneficial to both the clients and the business, proving a baseline of trust between the parties. This is especially true for business with little or no software security experience.

However, compliance is not always considered the best security improvement motivator for software. Regulatory requirements tend to be the lowest bar upon which all businesses are measured. Meeting the letter of the requirement does not mean that the software is truly secure, and such minimum effort often overlooks the spirit of the law. Some will even argue that regulatory compliance tends to narrow the focus of the security analysis, thus providing the business with a false sense of accomplishment. These compliance blinders may cause the business to potentially miss out on critical security issues not explicitly identified as regulatory requirements, or may cause additional challenges when the quality and specificity of the regulatory requirements is called into question.

Using compliance as a business motivator can also lead to miscommunication between the lines of business and the executive team. A recent study showed that while 54 percent of non-CEOs believed that compliance would decrease the risk of regulatory action, only 40 percent of CEOs believed this to be true.[8] Effective communication and alignment as to success criteria need to be clearly defined and measured.

Executive mandate

Establishing the culture for security is sometimes achieved by means of a specific executive mandate. The most notable of these executive mandates has been at Microsoft® with the implantation of Trustworthy Computing.[9] A mandate from no less than the chief executive officer stated, "Our software should be so fundamentally secure that our customers never even worry about it."

The success of using an executive mandate as a secure software motivation can be more readily understood when considered in light of funding and priority. If the organization is sufficiently funded to introduce software assurance, it can become a pervasive and essential part of all business service delivery. An initiative that is launched and mandated from the top of an organization also places emphasis on the business priority.

An executive mandate for software security needs to be tempered with the business need to provide solutions that increase the business value. Over-emphasis on security can lead to diminished innovation and a loss of focus on business goals. The introduction of security improvement must always be balanced with other aspects of the business.

Grassroots efforts

Finally, in contrast to the executive mandate, a grassroots internal effort can be a valuable approach to improving software security. Development team members are often motivated to improve software security shortly after taking an entry level security course or witnessing how simple it is to find and exploit a software vulnerability.

Understanding how to break and exploit software has an undeniable attraction for most developers. But an effective security awareness initiative requires a very different mindset from that of building software, since it requires team members to think in terms of "breaking" the software. Exploiting security vulnerabilities is a fresh and exciting new aspect of software design and development, and the growing interest in this can easily be harnessed to build more awareness and maturity among teams that build, assemble, and acquire business software solutions.

While the grassroots security approach is commendable, executive reliance upon this motivation is not without challenges.

Firstly, the effective transformation of entire business teams into security-proficient groups is extremely difficult. The time and effort to ensure that everyone is aware of the most current security threats and their edge cases is expensive. Rather than attempting to educate all employees on the details of security knowledge, it has been proved more effective to deliver a baseline of security knowledge to a core technical team, providing them with a solid understanding of the foundational requirements for delivering secure software. While this approach can work, it can also leave business leaders somewhat in the dark regarding security improvement planning.

Secondly, the lack of funding combined with the drive to deliver software on time and within budget often causes executive teams to neglect security until the last minute, frequently leading to "bolted-on" solutions as an afterthought. It is important for leadership to realize that the same requirements for software engineering that lead to high-quality software can also lead to more resilient and highly secure software. This is why security awareness and education is one of the most important first steps to any security program. At the very least, grassroots-led software security can be moderately successful through the use of security testing automation and well defined development processes.

Putting it all together

While none of the motivations described above should be seen as the single approach to improving software security, it is certainly possible to aggregate the best components from each.

Clearly understanding the cost benefit of finding and mitigating security issues early in the business life cycle can help to introduce ROI as a useful motivation. While compliance may not be an optional component in most organizations, it is important to recognize that no set of industry standards or regulatory requirements is likely to be comprehensive or prevent security incidents from taking place.

Of the four motivating factors for software security, communication from the executive team as to priority and funding are crucial for the success of improving software security. Finally, it needs to be understood that security is the responsibility of everyone within the business. No one is immune. Creating and fostering the culture of security through grassroots education and collaborative support teams is essential for software security improvement.

Understanding your environment: The evolution of software vulnerability

When it comes to software solutions, there are three core phases to its creation; design, development and delivery. The history of software vulnerabilities and safeguards against them actually runs in reverse order, since safeguards mostly responded to the actual attacks taking place and not to the potential threat associated with each of these phases.

Let me explain the three phases of this evolution.

Attacks based on the delivery environment

Attackers began to target the operational delivery of the software (1980 - 2000). These attacks could largely be averted by using network scanners, protocol scanners, configuration scanners, etc., so the success of these attacks was basically a function of whether or not the environment for software delivery had been properly configured. Over time, as firewalls, intrusion detection systems, and configuration management become the norm, the delivery attack surface became more and more minimized, forcing the perpetrators of malicious activity to focus on the software applications themselves.

Attacks based on the development environment

Thus, the next generation of automated assessment tools focused on the applications; for example, IBM Rational® AppScan® software is used by internal development teams to assess their own software for vulnerabilities. Ironically, similar assessment tools were used by attackers (2000 - 2010) to find exploitable vulnerabilities in the software itself. SQL injection and XSS not only became the most common vulnerability, but the most common attack "in the wild". These vulnerabilities are inherently propagated because the development teams, being unaware or insufficiently trained, introduced what I call "implementation vulnerabilities" during the development phase. Through better education practices, controls, and assessment techniques, the industry has begun to minimize the development attack surface-making these issues harder and harder to find. (We still have a ways to go.)

Attacks based on design

This brings us to the third category: design (2010+). It is not that the threats associated with the design of software were not present 10, 20, 30 years ago; it is simply that the attackers were not focused on these issues as there was far easier vulnerabilities to exploit from the delivery and development environments. Like all of us, the attacker has limited time and resources, and, in a twisted sense, they want ROI as well: "Where can I invest the least amount of time in order to exploit systems most efficiently?" Design is the next logical step.

So what types of issues are introduced in the design phase and how can we eliminate them? There are entire classes of problems that you can't run scanners to find, but which become logically evident if you stop to think about how an application is designed. Consider a system that has a "forgotten password" mechanism: it asks for your username and email, then sends you a new password. An attacker may simply be able to enter your username, with his own email address. Or consider software that uploads files to a file system that is not partitioned from the business environment. The attacker simply uploads a file that can be executed within this data-rich business domain. And so on. These are problems that take time and resources to uncover and exploit, but that certainly are important for the business to address.

Secure by design

This is where security solutions from IBM Rational really shine. The IBM Rational software development platform is completely focused on software design. From architecture management, to asset management, to requirements management—activities that dictate the security design of an application even before a developer touches the keyboard—the Rational software development platform helps address the following uncertainties:

- Is there an asset repository that stores all certified (secure) development components?
- Are there reusable security requirements that dictate the usage of these components as well as best practices?
- Are there architectural designs in the library that can easily be mined for the most secure product architectures?
- Can you prove through metrics that these components are being used pervasively?
- Does your software supply chain follow the same design practices that you do? How do you know?
- And more

Software development and delivery teams who plan for security early in the development phase find that testing can be done much more quickly and with more focus later in the development cycle. This not only gives increased efficiency, but it lowers resource costs and time to delivery. Here's a specific (and somewhat technical) example: If the security team certifies a specific code component for input validation and places it in an asset management system, the design team simply needs to add a development requirement that all user or system input is filtered through this component as part of the software specification. With this design requirement, we have not only eliminated buffer overflows, SQL injection, cross-site scripting (to an extent), command injection, and more, but we've also eliminated time-consuming assessment of input to the component and allowed the assessment team to focus only on where the development team did not follow this requirement. The time and effort (both human and automated) to track down these issues is greatly reduced. I could give many more examples—from architectures to authentication to authorization to logging—where "Secure by Design" efforts are a huge cost saving. Interestingly, these are simple best practices in modern software development, not specific to security.

IBM Rational provides the platform to allow these types of activities to take place, and to provide the assurance to the executive team.

For more information

To learn more about security solutions from IBM, please contact your IBM marketing representative or IBM Business Partner, or visit the following websites:

- Overview: "IBM: Web application security for a smarter planet"
- Solution brief: "Designing a strategy for end-to-end web protection"
- Podcast: "Web security overview"

General web resources:

- Web site compliance solutions
- Application security solutions
- Security and compliance management

Additionally, financing solutions from IBM Global Financing can enable effective cash management, protection from technology obsolescence, improved total cost of ownership and return on investment. Also, our Global Asset Recovery Services help address environmental concerns with new, more energy-efficient solutions. For more information on IBM Global Financing, visit: **ibm.com**/financing

About the author

Danny Allan is director of security research with the IBM Rational organization. Danny came to Rational through the acquisition of web application security leader Watchfire in July 2007 and has more than ten years of business and security technology-related experience. He currently participates in the IBM Security Architecture Board and most recently co-authored the IBM Secure Engineering Framework.

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
November 2010

Please Recycle

[1] IBM Security Solutions X-Force 2009 Trend and Risk Report

[2] IBM statistic, reported at RSA Security conference 2010

[3] IBM Security Solutions X-Force 2009 Trend and Risk Report

[4] Ponemon Institute 2009 Annual Report: "Cost of a Data Breach," Benchmark research conducted by the Ponemon Institute, at http://www.ponemon.org/local/upload/fckjail/generalcontent/18/file/UK_Ponemon_CODB%202009%20v9.pdf

[5] "Security in Development: The IBM Secure Engineering Framework," an IBM Redguide publication, at http://www.redbooks.ibm.com/abstracts/redp4641.html?Open

[6] I use the term "cost avoidance" deliberately, instead of "cost reduction." Security efforts almost always involve expenditures up front, but hopefully allow businesses to "avoid" more significant costs in the future. The net downstream result should be cost reduction, but it would be misleading to suggest that security solutions automatically reduce costs.

[7] See for example Barry Boehm and Victor Basili, "Software defect reduction top 10 list," IEEE Software, Jan. 2001 at http://www.cs.umd.edu/projects/SoftEng/ESEG/papers/82.78.pdf; and US Department of Commerce, Technology Administration publication, "The economic impacts of inadequate infrastructure for software testing," National Institute of Standards and Technology, May 2002 at http://www.nist.gov/director/planning/upload/report02-3.pdf

[8] IBM White Paper, Business Case for Data Protection: A Study of CEOs and Other C-level Executives in the United Kingdom, at https://www14.software.ibm.com/webapp/iwm/web/signup.do?source=swg-rtl-ponemonrptuk

[9] Bill Gates, "We can and must do better, " Trustworthy Computing Memo, at CNet, http://news.cnet.com/2009-1001-817210.html

RAW14238-USEN-00