WHITE PAPER

The Case for Building in Web Application Security from the Start

Sponsored by: IBM

Charles J. Kolodgy

February 2011

IDC OPINION

Over the past dozen years, the World Wide Web has transformed from a playground for technologists into a business platform. The Internet has become an economic juggernaut. IDC estimates that in 2009, 1.6 billion people used the Internet, with more than 624 million of those people making online purchases totaling nearly $8 trillion (both business to business and business to consumer). IDC estimates that ecommerce was close to $10 trillion in 2010. In addition to direct transactions, businesses use the Web for marketing, customer service, and information sharing. The enormous growth can be attributed to its ability to be a great equalizer for business. Companies of different sizes can compete on a level playing field. Small businesses reach a much larger clientele. Additionally, many companies are based solely online. This is all made possible because the primary elements of the Web are Web sites that anyone can create. It doesn't take many resources to design and deploy a Web site, and many, many companies produce their own Web applications. Unfortunately, the ability for anyone to create content is also the greatest weakness of the Internet.

There is a lot of badly written software resulting in software vulnerabilities that make Web applications inherently insecure. Some estimates state that at least 80% of all Web sites have one or more serious vulnerabilities. The conflict between Internet ecommerce positives and Internet insecurity has led companies, such as IBM, to offer products and services to improve Web application and Web services software. IBM's Secure by Design initiative encourages proactive IT security practices through the high-value opportunity of applying security throughout the software development life cycle. This improves Web application quality and security, thus making the Internet inherently more stable and secure. Improving the quality and security of Web applications also provides other benefits such as decreased costs in other forms of testing, increased awareness and proficiency in secure coding, and better organizational communication and collaboration on security issues.

IN THIS WHITE PAPER

In this white paper IDC highlights the growing IT challenge of protecting Web applications through software improvement and testing. It provides background on the problems associated with Web application insecurity, driving factors encouraging improved Web application security, and the means to validate the security of applications. It describes how IBM Rational's AppScan product line and IBM's Secure by Design initiative offer organizations the tools required to improve Web application security.

Global Headquarters: 5 Speen Street Framingham, MA 01701 USA P.508.872.8200 F.508.935.4015 www.idc.com

METHODOLOGY

IDC created this paper in December 2010. Its premises and opinions are based on leveraging a combination of research sources including IDC primary research on Web application security, historical and current research through IDC customer and vendor surveys, and information monitored on the subject of security as reported in blogs, the press, and other online information sources. In addition, IDC participated in briefings held by IBM in order to gain an in-depth understanding of IBM's application security products and business proposition.

SITUATION OVERVIEW

Undisputed Material Facts

When lawyers begin to litigate a case, they sit down to establish undisputed material facts in order to concentrate on more complicated issues. Looking at Web application security, we can also establish facts about software that are not debatable; specifically, that software is ubiquitous, critical, complex, and fallible.

- **Software is ubiquitous.** Software is all around us. It is everywhere because software is the closest thing we have to a universal tool. Nearly everything we touch either has software as part of it or was created or delivered with the assistance of a device that itself was run with software. The Internet itself is nothing but software and devices that run software. The face of the Internet is composed of Web applications that perform the multitude of tasks demanded by users.
- **Software is critical.** Computer scientist Bjarne Stroustrup, the father of the C++ language, has said, "Our civilization runs on software." He is completely correct on that count. We have already established that software is interwoven in much of our everyday activities. The Internet has become a critical cog in the information age. Over a billion people use the Internet and trillions of dollars in ecommerce transactions are spent every year, and this will only grow. It is estimated that at least 5 million companies depend on the Internet for the majority of their revenues. The Web is also critical for communications, outreach, marketing, and education. If you can think it, there is probably a Web application that can serve it up.
- **Software is complex.** Web applications, like all software, are complex. Software possesses a radical malleability that allows us to do with it what we will. Software itself is nothing more than a set of commands that tell a computer processor what to do. However, the set of commands can range from a few dozen lines of code to a million lines of code. Web functions are complex not just because of the raw lines of software code but because of the interconnections required for Web applications. Most applications are also dependent upon other code, and the interaction between applications adds to the complexity because you must trust the external inputs. On the Internet, the Web applications serve up content and transactions and interact with the user. Web application interactions are "smart" because just about anything is possible because the software makes it so. Web applications are the ghosts in the machine that give the appearance of intelligence when none can possibly exist. It is this appearance of intelligence that makes software so complex.

- **Software is fallible.** Critical applications that are complex are going to be fallible. Nothing is perfect, even though HAL from *2001: A Space Odyssey* thought it was. HAL stated, "The 9000 series is the most reliable computer ever made. No 9000 computer has ever made a mistake or distorted information. We are all, by any practical definition of the words, foolproof and incapable of error." However, it turned out that HAL had a flaw. It was programmed with two conflicting primary directives, and it couldn't reconcile this conflict. This is an example of a flaw, a weakness of the software architecture or design that creates a weakness in operation. The other type of software problem is referred to as a "bug." This is an implementation error within the software. An example of a bug for a Web application is not properly constraining the length of a user input.

Web Application Vulnerabilities

The dynamic nature of Web applications offers users unique experiences, but the technology that makes a Web site so interesting also has a dark side. People with malicious intent can turn this same technology against the enterprise to cause considerable damage, both to finances and to a company's reputation. As network security has improved, attackers are increasingly turning their attention to the application layer and the corresponding business applications that are running. The undisputed material facts mentioned earlier make it easy for attackers to go after an enterprise's critical assets, which are accessible through or within Web applications. Penetrating these applications will result in material benefits to the attacker, and given that software is flawed, there are direct avenues of attack.

The vulnerabilities created by Web application flaws and bugs are considerable. Some studies estimate that between 60% and 80% of Web sites have at least one serious vulnerability. Others contend that many Web sites have multiple critical vulnerabilities. There are a lot of these types of software problems. According to IBM's X-Force Trend and Risk Report, for the first half of 2010, over 55% of all vulnerabilities disclosed were Web application vulnerabilities, up from 49% in 2009. The number of vulnerabilities is ultimately related to how complex the code is and how much emphasis is placed on code quality. The majority of these vulnerabilities are design flaws and not specifically coding errors.

The total number of vulnerabilities wouldn't be such a problem if they weren't being exploited, but they are being exploited. IDC research indicates that at least 25% of all enterprises have been exploited through a Web application flaw. Additionally, many end users are being attacked by going to legitimate Web sites that harbor malware implanted in the Web site through a Web application vulnerability. Many people like the convenience of online shopping and banking, but many are wary of the risks associated with these online transactions.

There are numerous types of flaws and bugs within Web applications that can be exploited by attackers. The types of attacks vary in specifics but fall into classes of attacks. The vast majority of Web application attacks are associated with input fields. Web applications generally rely on user input, which is what makes them dynamic, but this input also opens them up to attack when the user input isn't contained. The following list highlights some of the most popular classes of Web application attack mechanisms. According to IBM's X-Force Trend and Risk Report, the most prevalent

categories of Web application attack techniques are cross-site scripting and SQL injection. Over 60% of all attack types fall into those two categories.

- **Cross-site scripting.** Cross-site scripting attacks allow for the embedding of malicious code into a page the user is visiting. This malware can manipulate the behavior or appearance of the Web page, hijack the user's browser, implant software, or redirect the user to another site. These attacks are made possible whenever an application takes untrusted data and sends it to a Web browser without proper validation and escaping.

- **SQL injection.** SQL injection can occur when untrusted data is sent to an interpreter as part of a command or query. Specifically, the attack uses a standard input vector (like a form field) to inject SQL commands that can be executed by a database. A successful SQL injection gives an attacker access to a back-end database, which can allow the attacker to steal, modify, or destroy confidential information or even embed other attacks inside the database. This attack is possible only when user input is insufficiently or improperly validated.

- **Application buffer overflow.** Buffer overflow attacks occur when very large amounts of data are included in an input field. If the request exceeds the allocated buffer size, attacker code (which is also embedded in the input field) will then be executed, effectively taking over the application. The most effective method to defend against these attacks is to ensure that the application provides appropriate size checking on all such inputs.

- **Authentication bypass.** Errors in the application functions can allow attackers to circumvent the authentication of the application. This can compromise passwords or session tokens or allow attackers to assume other users' identities.

- **Forceful browsing.** A very simple attack is forceful browsing, which involves modifying a URL in the attempt to bypass Web controls in order to break out of a server's root directory and access files on the rest of the file system. Since almost all input to a Web application is from a browser, anyone can attempt forceful browsing. Extra authorization at every entry point into the application will fix this problem.

- **Cookie poisoning.** Cookies are used more and more by Web applications, so when an attacker manipulates a session cookie, he or she has the opportunity to obtain unauthorized information from the server.

- **Cross-site request forgery.** This attack method is one that pulls together a number of attack types, allowing an attacker to perform functions that are generally authorized. This attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable Web application. This allows the attacker to force the victim's browser to generate requests that the vulnerable application thinks are legitimate requests from the victim. Successful cross-site request forgeries have been used to take monies out of bank accounts.

 ©2011 IDC

It can't be reiterated enough that almost all of these attack methods are made possible only through flaws in the Web application code. Almost all of them are tied to errors in accepting user (i.e., browser) input. Developers have an excessive trust in the data being processed. These attacks are made possible because the input (which is generally untrusted) isn't properly validated or the application isn't constrained to accept only data of a specific type or length. To be fully secure, enterprises need to be able to remove the vulnerabilities these attacks exploit. Almost all of these vulnerabilities can be identified and remediated early in the Web application development process, provided security design and testing is a component of the software development life cycle (SDLC).

Demanding Better

The squeaky wheel is the one that gets attention, and for enterprise cybersecurity, the noise is coming from regulatory and industry mandates. In some markets, 80% of security spending can be directly tied to compliance. Web application security isn't to that level, but regulations mandating improved software security are on the horizon. The International Organization for Standardization (ISO) has standards (ISO 27001 and ISO 27002) that provide some basic coverage for "developing and maintaining in-house software." However, ISO standards are best practices. A specific standard mandating improved Web application software is the Payment Card Industry Data Security Standard (PCI DSS). PCI DSS already includes a requirement that merchants "develop and maintain secure systems and applications." This requirement includes vulnerability assessment, patching, change control, the integration of security into the software development life-cycle process, the use of Web application development best practices, and code reviews for custom-developed, Web-facing applications.

The payment card industry isn't the only group pushing to mandate secure software, especially custom software such as Web application. Security trade and training organizations such as OWASP and SANS are encouraging organizations to require that Web application security be identified within requests for proposals and ultimately within contracts. A number of states have taken up this cause, just as they have done regarding public disclosure of data breaches. Nevada has adopted PCI DSS as its security standard. States like New York and Oregon are encouraging that state procurements of custom software include provisions mandating that vendors "identify the tools to be used in the software development environment to encourage secure coding." The requirements also require documentation to demonstrate that code reviews and vulnerability and penetration testing are part of their SDLC. Some states are trying to encourage that Web applications be, at a minimum, scanned against the top 10 or 25 vulnerabilities as identified by SANS or OWASP. The federal government is also getting into this effort.

The Federal Information Security Management Act of 2002 (FISMA), along with its NIST enabling standards (SP 800-53, FIPS 199, and FIPS 200), governs the management of information security in the federal government. Code scanning isn't a specific requirement, but the standards do require regular security assessments and review, which can be accomplished through Web application scanning. A federal mandate that would require Web application scanning is the Lieberman-Collins bill, Protecting Cyberspace as a National Asset Act of 2010 (S.3480). This legislation

would reform government procurement by "creating a system that requires acquisition officers in the federal government to have the knowledge that they need about the vulnerabilities in products." Internationally there are similar regulations and standards that require improved security, including Web security. The Basel II international banking standard calls for improved operational risk management, which requires improved IT security. Various privacy laws in countries such as the United Kingdom (Data Protection Act) and Japan (Personal Information Protection Act) mandate that reasonable actions be taken to protect personal information from disclosure, unauthorized use, or destruction. These requirements expand the need for improved application security. Additionally, ISO 27001 and ISO 27002 standards provide best practices to include vulnerability scanning.

This wheel is beginning to squeak, and software developers and business managers need to prepare now to address mandates that will be coming. IDC expects that many organizations, including large corporations, will begin to require improved software security from their custom software providers. There is a trend of pushing security responsibility toward the application vendors. IDC expects that the ability to deliver secure code will become a key differentiator between software providers.

SOLUTIONS

Attack Cause, Not Symptom

The vast majority of security comes from solutions that are bolted on to a network device or application. These workloads include things such as firewalls, intrusion prevention, authentication, and encryption. All of these perform specific tasks that are required, but in reality they are able to deal with only the symptoms, not the root cause of insecurities. Security professionals are realizing that the weakest links in the Web security chain are Web servers, Web applications, and related back-end databases. As exploits associated with Web applications continue, IT management is coming to the conclusion that insecure coding is the root of many a breach — one might even say source code is the source of many security woes. The need to address the problem by developing applications with security in mind to remove security defects from applications prior to release is gaining adherence.

Producing more robust Web applications at the code level closes potential doors that attackers can use. Attack-resistant Web applications contribute to the overall security posture. The external security solutions can then be used in a targeted manner, making them ultimately more effective. The bottom line is that the most effective way to secure custom-developed Web applications is at the software level. Software is the root cause of most Web vulnerabilities; therefore, removing the software flaws will make the applications inherently resistant to attack, thus reducing the reliance on external, bolted-on security.

Measure Twice, Cut Once

In carpentry you are told to measure twice before you cut a piece of wood to ensure that it is the length required. For Web application security testing, measuring twice comes from the two types of software security testing: static and dynamic.

Static testing, also called source code scanning and white box testing, analyzes the software source code or binaries to gain information about the software. Static testing looks at the software at rest. The process can provide an understanding of the code structure and can help to ensure that the code adheres to proper standards. It is effective at identifying flaws in code functionality and syntax errors. The use of automated static source code scanning helps teams focus on high-risk areas for deeper manual code reviews.

Dynamic code analysis is usually referred to as black box testing. Here the running software is examined to see how it behaves under different input. These tests do not require specific knowledge of the application's code structure, and programming knowledge is not required. The goal of dynamic testing is to test the functionality of an application. It looks at what the application will do with both valid and invalid input. This testing is interested in what an application will do that isn't expected. As a security tool, it simulates an external attack.

These two testing types are best used in concert as part of a complete SDLC security program. Static analysis is only a first step because automated source code analysis has difficulty in identifying logic errors, timing errors, resource conflicts, data errors, and configuration errors. Manual source code reviews and dynamic testing are best at finding those errors. Additionally, dynamic analysis is often performed in an effort to uncover subtle defects or vulnerabilities.

People understand the need to perform functional testing to ensure that applications provide the services expected, but often overlooked is testing to ensure that applications do not include unintended operations, many of which become security vulnerabilities. This is more critical in Web applications because of the way they receive and process unstructured user commands. The Web application has no control on what the user inputs; rather, it has control only on the output. A key to the dynamic security software testing is to uncover unexpected, unintended, undocumented, or unknown functionality. By testing in this manner you can reduce the attack surface. Security testing tools alone do not make software secure, but as part of a software security program, they do help reduce the attack surface.

Process

Developing high-quality and successful software in the 21st century requires a paradigm shift. Rather than adopt the 20th century attitude that software is made to solve problems and people will use it for the greater good, developers need to take the position that their software will be attacked the second it is deployed and they must build applications that can defend themselves in a hostile environment.

Web applications are too important to be created in a haphazard manner, and secure Web applications do not happen by accident. They happen when every developer, tester, and manager on a project takes security seriously and builds in testing during every phase of the software development life cycle. Security is not something that is addressed at the end of a product cycle, nor is it a specific milestone that occurs during project execution. Security must be on everyone's mind throughout every phase of the software life cycle. A misstep at any phase can have severe consequences.

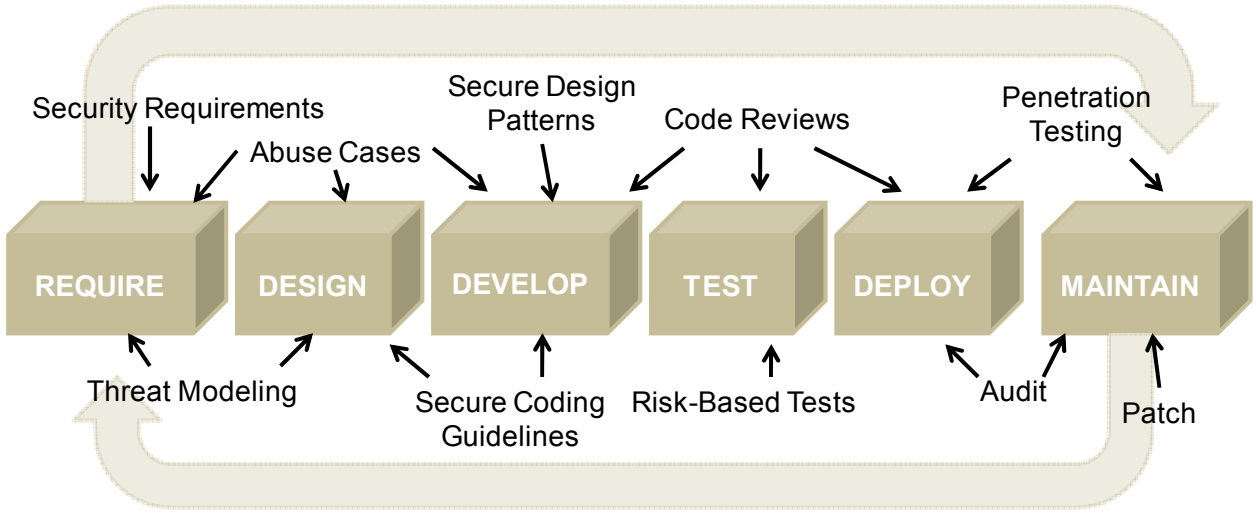
This means secure software is a software development problem. Every member of the software development team — from managers and support staff to developers, testers, and IT staff — must consider software security an integral part of the application development process, not an afterthought. An effective development process assumes that code will be attacked and embeds checks at every iteration and/or phase of the process. Iterative processes are better because they offer more assessment points and allow for the capture and remedy of security vulnerabilities early in the SDLC.

Making this work requires process improvement and executive commitment. Security considerations must be explicitly worked into the entire SDLC. Security is part of the requirements process to determine what security functionality should be included, just as the overall functionality of the application is being created. In the design phase, risk analysis must be included to uncover and rank risk so that mitigation of those risks can start immediately and not wait until deployment. Security testing with static analysis is part of the coding process to discover software bugs during code creation. Closer to application deployment, dynamic penetration testing is conducted to get a real-world view of the security of the application. Figure 1 illustrates how the security elements fit into the software development process.

Ultimately making security a part of the SDLC will require training. Software developers and quality control staff will need to be trained to improve their security awareness. They will need training on security application vulnerability scanners, and they will need to be provided with advanced software security design principles and risk mitigation techniques.

FIGURE 1

Security Software Development Process



Source: IBM, 2010

 ©2011 IDC

Benefits

For many, security is its own reward. In a survey, IDC asked participants how they measure the business value of security, and a large segment of answers fell into the category that security is a must-have. However, IDC believes that security for security's sake doesn't provide a business case. IDC believes that enterprises engaging in Web application security improvement should be looking for much more, and there are considerable benefits to including security within the Web application SDLC.

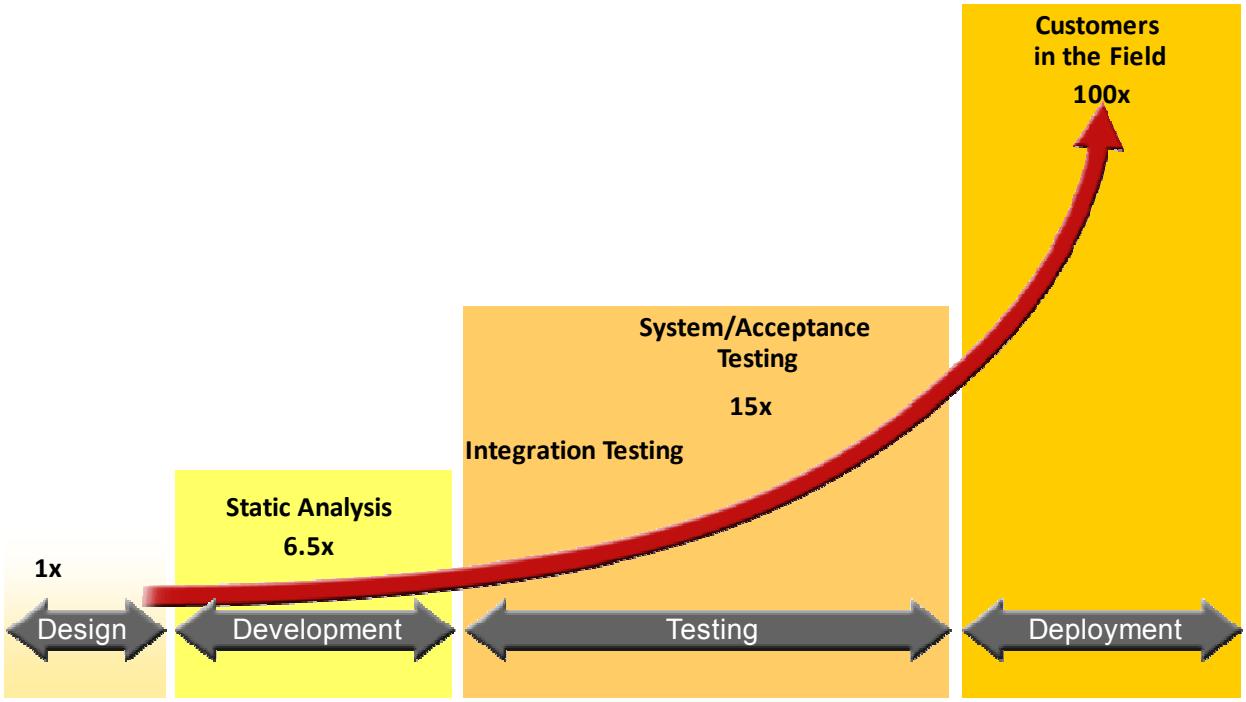
The primary benefit is that secure software is cheaper to maintain than flawed and buggy software. There have not been many recent studies on the costs associated with software security, but there is data on software quality. This fits well because software security is a subset of software quality. In May 2002, NIST published a report called *The Economic Impacts of Inadequate Infrastructure for Software Testing*. The report stated that buggy software may be costing the U.S. economy $59.5 billion a year. That study, along with a 1981 IBM Systems Sciences Institute report entitled *Implementing Software Inspections* and Capers Jones' 1996 book *Applied Software Measurement*, provides similar statistics on the value of identifying and correcting software defects early on during the software development process. On average it costs 15 times (15x) more to repair software at testing than in the design phase. It can cost up to 100 times (100x) more to rework deployed software (see Figure 2). Recently, Jones revisited the cost issue in his paper *A Short History of the Cost per Defect Metric* (May 2009). He looked at the metrics differently. The cost-per-defect metric can distort the calculation, but he also believes that software quality has value due to the reduction in defect repair costs and the reduction in development and maintenance costs.

All of this data on the additional cost associated with fixing software problems (both security and quality) marries up well with data from another 2002 survey of software that reported that 70% of security defects were design flaws and that nearly half of those could have been fixed inexpensively in the design phase. A consensus number provides us with a final comment that software development teams spend, on average, almost 40% of their time on rework. By improving the quality of software developed, enterprises can realize great savings.

This data would appear to be dated, but there is no indication that these statistics are any different today. The economics of building quality and security software should not be radically different today than it was a decade ago. The bottom line is that secure software is going to be cheaper to maintain.

FIGURE 2

Cost per Defect Metric



Source: IBM, 2009

Software developers must understand that the benefits will not accrue immediately. There will be short-term costs associated with the purchase of Web application vulnerability scanning tools and secure coding training for developers and software engineers. Projects initially will take a little longer to deliver because of the learning curve, but in the long run, those times will come down. Maintaining software will be less expensive because more secure code should be more reliable, thus reducing patching and production system downtime. In the long run, software development times will improve because of the knowledge gained during testing and implementation of best practices. As has been stated earlier, all this is especially true the earlier security is incorporated into the SDLC. Errors are much easier and much less costly to repair earlier in the process. In summary, specific benefits of including security in the SDLC are noted in the following list, but the main strategic benefit is that development teams will be more competitive.

- Reduction in software vulnerabilities
- Ability to meet compliance requirements
- Reduction in costly rework by detecting and eliminating flaws at the earliest possible opportunity
- Intangible benefit of improving developer job satisfaction by allowing developers to create more and rework less

 ©2011 IDC

- Security culture ingrained as quality and reliability
- Reuse of trusted software in future development
- Improved customer experience

IBM SECURITY SOLUTIONS: SECURE BY DESIGN

Security must be everywhere. It should begin at project inception and be on the mind of every engineer during requirements analysis, design, coding, testing, and deployment. This is the only way that security can be reliably improved in every product. This is the differentiating philosophy IBM has adopted with its Secure by Design initiative. Secure by Design means that security is intrinsic to all business processes, including product development, daily operations, and business planning. Specific to application security, by having security "at the table" from the beginning, instead of trying to place the square peg of security into a round hole representing a developed product, organizations can securely and safely adopt new forms of technology. Cloud computing, virtualization, collaboration, and advanced Web 2.0 applications can be more safely leveraged for cost benefit, innovation, and shorter time to market.

In the area of Web application security, IBM provides security and application development teams with the tools to scan, identify, and prioritize Web application security risks in preproduction applications to help ensure the development of secure code. They also have solutions that help improve the security of existing, fielded Web applications. The IBM Rational AppScan product line includes the following:

- **AppScan Standard Edition.** This is the flagship product. It provides automated Web application security testing for the whole development testing team — IT security, auditors, and penetration testers. While it has been known as a dynamic analysis solution, it has recently added static analysis testing to its capabilities. It scans Web sites for embedded malware and links to malicious or undesirable sites and also tests for vulnerabilities in Web services. AppScan Standard Edition has regulatory compliance reporting templates with 40 out-of-the box compliance reports.
- **AppScan Source Edition.** The Source Edition of AppScan is a static code scanner. It provides automated security testing of the source code during the early stages of the application life cycle. It can pinpoint vulnerabilities and provide precise, detailed remediation advice for rapid fixes within the IDE for development teams.
- **AppScan Build Edition.** This version embeds security testing into the SDLC to find vulnerabilities in the build environment. It combines static and dynamic analysis techniques for complete scanning coverage. It provides intelligent fix recommendations to improve productivity by offering developers suggested fixes to discovered flaws.

- **AppScan Tester Edition.** This product is used by quality assurance (QA) personnel. It automates security testing within the quality assurance environment. It has full integration with IBM Rational Quality Manager software. In this way, it supports standard QA workflow activities, including invoking tests, working with results, prioritizing and submitting defects, and performing regression testing.

- **AppScan Enterprise Edition.** This product enables organizations to facilitate communication and collaboration between information security, development, and management. It enables organizations to engage and educate their development and QA teams and implement security controls throughout the SDLC. It provides management with visibility into the security and regulatory compliance risk of their Web applications.

IBM Rational's AppScan is a strong product line that provides both static and dynamic scanning with multiple delivery methods to serve any potential client. However, AppScan is only the surface representation of what IBM Rational can provide in assisting an SDLC security strategy. Many other critical software security components have been incorporated directly into the Rational product lines. Developers and software engineers can utilize these features, such as security knowledge base and templates or object reuse of validated components, to intrinsically improve software security and reliability. Web application software security isn't built on scanners alone, IBM agrees.

IBM not only presents Secure by Design to its customers but also uses it internally. Security is intrinsic to IBM's business processes as well as its product development and daily operations. It is factored into the initial design, not bolted on after the fact. IBM has additional products that complement the dedicated application security products. Examples are IBM security identity and access management solutions that can be used to authenticate users within applications and establish specific roles that adhere to policy. IBM security intrusion prevention solutions offer virtual patching and Web application firewall for added security. They also have AppScan integration that allows customers to customize Web application protection policies based on AppScan's intelligence in order to proactively protect existing vulnerabilities.

CHALLENGE: PROPER INCENTIVES

Everyone involved in Web application development strives to make robust and secure software. However, a want doesn't make it a reality. Two prime barriers make it difficult for developers to produce secure code. The first is they may not have the knowledge or skills to write secure code, and the second is they are generally not incentivized to produce secure code. The first issue can be solved with training. There are a growing number of training opportunities on secure Web application development. Additionally, and probably more accessible, is the knowledge base available within IBM Rational AppScan. The intelligent fix recommendations within AppScan provide the background on why specific coding is vulnerable and provides strong coding suggestions on how to remediate the vulnerability.

The issue of incenting developers can come only from management. In most software development, the most important metric is delivering an application within a certain time frame. The software is to meet the functionality requirements, but there is generally no requirement on how robust the software is at thwarting attacks. To encourage developers, managers and executives must elevate the importance of software security. The mindset of "release it now, fix it later" has to be changed to "we will ship no code before its time" to paraphrase Orson Wells. The easiest way to do this is to reward developers who produce secure code instead of penalizing them when deadlines slip. Managers can use the reporting capabilities with the AppScan family of products to get an understanding of where the development team is regarding security and reward those with the least number of defects. By changing what is measured and rewarded, enterprises can help secure coding become a reality.

CONCLUSION

It is not hard to agree on the undisputed material facts that software is ubiquitous, critical, complex, and fallible. Few can dispute the power of Web applications to improve business processes and to offer expanded and competitive opportunities. The dynamic nature of Web applications offers users unique experiences. Long gone are the days of Web sites being static electronic information brochures. Now they conduct critical business functions, such as brand loyalty, customer support, and commerce. Web sites provide the complex and rich experience that users have come to demand. For a vast number of companies, their Web presence is a major asset. However, the dark side of this technology is that online outlaws are working overtime to exploit any security hole they can find in Web applications. Deployed software is continuously under attack.

Only a concerted effort by the software development community to produce more robust and reliable applications will foil attackers and allow users and stakeholders to feel confident that they are protected from exploitation.

There is too much at stake to take a cavalier attitude toward Web application security. Software development teams need the resources necessary to ensure their software is secure. The best way to accomplish this is to incorporate security into the software development life cycle. Developers and quality assurance staff should be responsible for security, just as they are responsible for functionality and quality. Vulnerabilities in custom Web applications need to be discovered and resolved in the most efficient manner possible. In the long run, incorporating security as a quality element within the SDLC has considerable benefits, which can include cost savings resulting in less rework of nearly completed or fielded software. Additionally, IT customers are becoming increasingly concerned about vulnerability risk profiles; thus, they are evaluating the effectiveness of their software vendor's security assurance programs. IDC believes that software vendors that pay careful attention to secure development and deployment practices, investing the time and effort to build security in from the ground up, stand to benefit the most from this overall trend.

A major component of security within the SDLC is vulnerability testing tools. Without automated vulnerability testing, it is extremely difficult to secure complex Web sites. The tools aid in discovering security-related flaws and bugs so that they can be eliminated before a program becomes operational. Errors are much easier and much less costly to repair earlier in the process.

IBM, with its application security products and Secure by Design philosophy, can be a trusted partner. IBM understands the need for security and believes the solution should be an integral part of the whole, instead of just something appended after the fact. Its experience in software development, quality, and security means it can provide assistance throughout the entire SDLC. When selecting a Web application code scanner, entities should consider how it integrates into the SDLC and its ability to accurately discover vulnerabilities associated with multiple attack methods, as well as whether it can handle both security and policy compliance issues, is accurate, provides a knowledge base for remediation, and has central reporting capabilities. IBM Rational AppScan can meet all those requirements, so it should be considered when looking at this technology.

Copyright Notice

External Publication of IDC Information and Data — Any IDC information that is to be used in advertising, press releases, or promotional materials requires prior written approval from the appropriate IDC Vice President or Country Manager. A draft of the proposed document should accompany any such request. IDC reserves the right to deny approval of external usage for any reason.

Copyright 2011 IDC. Reproduction without written permission is completely forbidden.