



WHITE PAPER

Managing Application Security in Business Processes



Where it all comes together.™



CONTENTS

+ Introduction	3
+ The Issue	3
+ Procedural Strategies: The Development Lifecycle	4
+ Vulnerability Identification	5
Separate Development and Quality Assurance Environments	5
Patch Distribution	5
Temporary Countermeasures	6
Failure Modes	6
Change Control and Management	6
Additional Measures: Policy and Developer Awareness	6
+ Technical Strategies	7
Assessing Web Application Vulnerabilities	7
Implementation of Logging	9
Implementation of Monitoring Technologies	9
+ Conclusion	10



Introduction

As business-process automation started to take hold in the early 1990s, organisations began to replace people with mainframe applications and EDI transfers to perform mundane tasks including data entry and processing. However, for crucial business processes such as wire transfers, customer database queries and purchase orders, organisations continued to use human intervention, believing that auditor supervision was required to ensure accurate money transfers and appropriate access controls to sensitive information.

Today, no division of labour exists between the tasks for which applications and people are responsible. Conversely, human auditors and applications work together in concert to manage business processes and the Web servers, databases and middleware on which they depend. However, many of these applications, especially Web-based ones, are rife with vulnerabilities, ranging from SQL injection to cross-site scripting. Even the platforms they run on are far more vulnerable than their predecessors, mainframes and leased-line transfers. As a result, although applications help expedite business processes, at the same time they expose organisations to a considerable amount of security risk compared to human auditors.

Ensuring the correct functioning of these applications, and, by extension, the business processes they support, has become crucial to an enterprise's success, and managing application vulnerabilities has thereby grown vastly in importance. This paper will clarify the issue of application vulnerability management and provide high-level strategies to mitigate the risks those vulnerabilities pose to business processes.

The Issue

For the purposes of this paper, a business process is a set of actions facilitating the transaction of business with an external or other internal entity. Although using information systems to automate business processes is much more efficient and cost effective than employing human auditors, it has consequences that have yet fully to be appreciated. Organisations that value time to market over security often automate business processes using Web applications. In many cases, these organisations do not establish the strong security controls and auditing functions needed to mitigate the risks, which can result in serious reputation and financial damage.

In the past, enterprises relied exclusively on human auditors to maintain controls by examining the following:

- Regular operation: is the business process functioning properly?
- Failure modes: what happens when the business process or one of its components malfunctions?
- Logging and auditability: what information does the business process record about itself and can the business process be reconstructed after the fact?

The Sarbanes-Oxley Act, recent US legislation that affects every publicly traded company, mandates that auditability controls must be in place for financial transactions.

The information systems now accountable for business functions must also assume responsibility for both the technical and procedural functions of the human auditor. The remainder of this paper is divided between procedural and technical strategies for application-vulnerability mitigation.

Procedural Strategies: The Development Lifecycle

As previously mentioned, business processes and the interconnected systems that support them now rely on Web applications, many of which have not been adequately tested for security vulnerabilities. Generally, a process-based, proactive approach is highly effective. An overall strategy for business-process protection will include a well-described inventory system for critical business processes, diligent monitoring programmes to detect abnormal activity and a detailed process for managing the 'content' for detective systems, such as intrusion-detection system (IDS) signatures.

For application-vulnerability management in particular, the following procedures are paramount:

- Mechanisms to identify the specific vulnerabilities of systems supporting critical business processes
- The ability to perform testing in a non-production environment and to cancel unsuccessful changes
- Mechanisms for distributing patches on a regular basis with reasonable operational costs and impact
- The ability to introduce temporary countermeasures when patching cannot occur on a timely basis
- Planning for failure modes (ensuring that damage is minimised if something goes wrong with the application)
- Compliance with regulatory requirements
- Appropriate change controls and management
- Proactive measures, such as policy and developer awareness

All these procedures should be placed within a unifying framework, commonly known in this case as a development lifecycle. Security checks should be incorporated, ideally as gateways to successive phases. In addition, the auditing and logging functions should be integrated into the development process early on so that automated tools, scripts or applications can meet the requirements of the enterprise as well as applicable regulations.

As previously mentioned, business processes and the interconnected systems that support them now rely on Web applications, many of which have not been adequately tested for security vulnerabilities. Generally, a process-based, proactive approach is highly effective. An overall strategy for business-process protection will include a well-described inventory system for critical business processes, diligent monitoring programmes to detect abnormal activity and a detailed process for managing the 'content' for detective systems, such as intrusion-detection system (IDS) signatures.

Vulnerability Identification

From an application perspective, vulnerability identification is absolutely critical and often overlooked as a source of risk. Unverified parameters, broken access controls and buffer overflows are just a few types of the many potential security vulnerabilities found in complex business applications, especially those developed internally. (For strategies to approach these problems proactively, see Additional Measures: Policy and Developer Awareness, on page 7.) Unfortunately, commercially developed applications are often equally insecure, therefore requiring a vigilant patching process.

It is often said in information-security circles that an application is only as secure as the platform on which it is running. Formulating and documenting secure builds for each operating system should be done on an enterprise or group level. Deviation from those standards should require compensatory controls and this compromise should be reviewed as part of the development lifecycle. Production servers and other systems should be frequently scanned using the latest signatures to ensure their immunity from system-level vulnerabilities.

+ Separate Development and Quality Assurance Environments

Development and quality assurance (QA) environments must be separated from the production network, ideally by a firewall. It is also possible to use VLANs or a hardened router, but because development boxes are often configured insecurely, they should be removed from the production environment. This step is even more important when the production environment is a demilitarised zone (DMZ) accessible from the Internet.

Before any application that supports a business process is promoted to a production environment, it must be tested for known vulnerabilities at both application and system levels. This is generally regarded as the final step in the development lifecycle and should be repeated when an application is patched, reconfigured or upgraded on the operating system.

+ Patch Distribution

Both operating systems and applications developed by other companies must constantly be monitored for the latest available security patches. Developers should also examine the code of applications created internally and release patches or new versions as needed. Maintaining this degree of vigilance is difficult for large organisations with high uptime requirements and complex environments. The central criterion for success of a realistic patch-distribution process is whether it can be applied before an exploit (automated code designed to take advantage of a given vulnerability) is posted on the Internet. As mentioned above, patches must be tested in the QA environment before they are applied.

+ Temporary Countermeasures

An agile enterprise is able to implement temporary countermeasures, especially when a rampaging worm or virus makes it impossible to wait for thorough testing and careful application of patches. Often, this strategy means following a defined incident-response plan and shutting down ports on firewalls and routers or blocking IP addresses. The ability to put together an incident-response team, notify all necessary parties (internally and externally), and follow clear guidelines for escalation of issues is integral to successful implementation of these countermeasures.

+ Failure Modes

A failed application can have a negative impact on a business process, especially if it goes undetected. Given the many ways in which an application can break down, an organisation must build in multiple checks during the development process to avoid failures proactively. Nonetheless, the possibility of a failure cannot be completely eliminated. For an organisation to manage its risk potential effectively, it should devote a step in the application and maintenance process to understand failure modes during testing as well as the operational difficulties encountered once in production.

When an application fails, the organisation should have provisions in place for the overall business process (that is, a rerouted transaction that does not depend on the application or the means to perform the application's functions manually), but, most importantly, an application should never 'fail open'. This phrase refers to an instance when, upon failure, all types of transactions are allowed through the system. For instance, when a firewall rule set fails, it allows all traffic of a given type to pass. An application or its internal checks should always fail closed.

+ Change Control and Management

Many corporations have developed various ad hoc methods of transferring financial data to and from their business partners. Much of this data includes profit-and-loss projections, contractor invoice submissions and other sources of questionable integrity because of the way they are transmitted and the insecure nature of the systems on which they are stored. In other words, there is no guarantee that the data is unmodified or has not been removed between the time it was sent and received. Furthermore, the lack of a secure transfer may hamper cyber-forensics or other responses to computer security incidents if thorough change-control and change-management procedures are undocumented and not scrupulously followed.

A clear understanding of how an application that supports a business process has changed is the only way to reconstruct a transaction days, weeks, or even months after it has occurred. When reconstructing a transaction, organisations are required to comply with the Sarbanes-Oxley Act, which mandates that both the corporate officers and an external auditing agency attest to the belief that the financial data is being handled in a manner that preserves its integrity.

Every corporation has some standard business processes involving systems that contribute to data manipulation and storage. These systems can include mainframe databases, Web-facing applications, business-to-business processes, and third-party connections, among others. Thus, all presentation, application and data tiers of the financial system architecture that handle money and touch a business process must have sufficient security controls in place to obtain a Sarbanes-Oxley 404 certification.

+ Additional Measures: Policy and Developer Awareness

It is essential to educate developers properly about the tools and techniques required to write secure code as well as the importance of the project. Policies regarding secure code should be kept at a level high enough to meet the needs of the full breadth of the enterprise's development projects.



However, note that developers will often disagree about the best method to achieve security in any given piece of code and that flexible guidelines may be developed for more specific guidance. Peer review is also a useful part of the development lifecycle.

A developer writing code that will have an impact on a business process should possess a basic knowledge of the topic, which is outlined in the following diagram:

Development Lifecycle Process

Requirements Definition	Design and Develop	Test	Implementation
Identify business needs	Design test plan	Execute test plan	Execute implementation plan
Identify confidentiality, integrity, and availability requirements	Execute design	Secure approvals for test-plan results	Conduct end-user training
Develop functional and technical specifications	Develop system documentation and update process	Return to develop stage if unsatisfactory and retest	Postimplementation reviews
Develop change-control strategy	Develop contingency back-out plans	Secure final approval and go-live decision	
Secure approvals	Create training materials		

The following section will provide additional detail on the first two stages listed in the table above.

Technical Strategies

For existing applications, several technical strategies exist to manage vulnerabilities. The first is an assessment methodology to eliminate Web application vulnerabilities, and the second is to ensure that logged applications are compliant with relevant regulatory requirements. The third is the logical implementation of monitoring technologies.

+ Assessing Web Application Vulnerabilities

Three basic ways exist to assess Web application vulnerabilities: assign internal personnel or hire a security consulting firm to conduct manual testing, or use a software product to perform automated tests. For most companies, including large organisations, it is not cost effective to cultivate the highly specialised skills internally that are needed to perform assessments of Web application vulnerability. The other two approaches, which have proven effective in a large number of enterprises, are discussed further below.



Manual Testing

The preferred solution in addressing Web application vulnerabilities to date is manual testing. Security consulting firms train specialised personnel to review Web applications from a black-box or white-box (code-review) perspective and the consultants are able to capitalise on their experience with the developers' common mistakes. This method eliminates a large proportion of security flaws, as shown in the table below. For more complex applications, security consultants interview developers, assess connections to other applications (and session management across them) and review the entire architecture of the application for components such as adherence to regulatory requirements (often regarding encryption) and service-level agreements with third parties. This is still the preferred approach for enterprises, such as financial institutions or manufacturing firms that cannot tolerate security compromises in their applications.

Introduction to Information Security and Privacy	Secure Web-Application Development	Secure Configuration Management	Final Presentation
<ul style="list-style-type: none"> Overview of information security and privacy concerns Overview of common attack techniques and tools Benefits of effective controls Sources for additional information 	<ul style="list-style-type: none"> Secure application development Secure database design Operating system and platform considerations Network security considerations 	<ul style="list-style-type: none"> System development lifecycle Secure baseline configurations Change control and management Security testing in QA 	<ul style="list-style-type: none"> Instructor presentation Student presentation In-class example

Top Vulnerabilities in Web Applications		
A1	Unvalidated Parameters	Information from Web requests is not validated before a Web application uses it. Attackers can use these flaws to attack backend components through a Web application.
A2	Broken Access Control	Restrictions on what authenticated users are allowed to do are not properly enforced. Attackers can exploit these flaws to access other users' accounts, view sensitive files or use unauthorised functions.
A3	Broken Account and Session Management	Account credentials and session tokens are not properly protected. Attackers who can compromise passwords, keys, session cookies or other tokens can defeat authentication restrictions and assume other users' identities.
A4	Cross-Site Scripting (XSS) Flaws	The Web application can be used as a mechanism to transport an attack to an end user's browser. A successful attack can disclose the end user's session token, attack the local machine or spoof content to fool the user.
A5	Buffer Overflows	Web application components in some languages that do not properly validate input can be crashed and, in some cases, used to take control of a process. These components can include CGI, libraries, drivers and Web application server components.
A6	Command-Injection Flaws	Web applications pass parameters when they access external systems or the local operating system. If an attacker can embed malicious commands in these parameters, the external system may execute those commands on behalf of the Web application.
A7	Error Handling Problems	Error conditions that occur during normal operation are not handled properly. If an attacker can cause errors to occur that the Web application does not handle, they can gain detailed system information, deny service, cause security mechanisms to fail or crash the server.
A8	Insecure Use of Cryptography	Web applications frequently use cryptographic functions to protect information and credentials. These functions and the code to integrate them have proven difficult to code properly, frequently resulting in weak protection.
A9	Remote Administration Flaws	Many Web applications allow administrators to access the site using a Web interface. If these administrative functions are not very carefully protected, an attacker can gain full access to all aspects of a site.
A10	Web and Application Server Misconfiguration	Having a strong server configuration standard is critical to a secure Web application. These servers have many configuration options that affect security and are not secure out of the box.

Automated Testing

Automated testing has matured in the last few years. Many products in the marketplace do an excellent job of identifying common vulnerabilities, but they are generally expensive and cannot assess an application dependent on complex layers of middleware and, in some cases, third parties.

+ Implementation of Logging

Regulatory requirements and the maturation of correlation technologies have driven organisations to devote the appropriate resources to implement logging on all systems and supporting business processes properly.

Standard implementations of 'syslogs' have created a baseline for gathering normalised data, but processes for reviewing and correlating the logged information have lagged slightly behind. Today, providers of Managed Security Services (MSS) offer correlation services, security software vendors sell correlation engines, and, per the Sarbanes-Oxley Act, logging is required of financial transactions for all public companies. Security-relevant logging information should be reviewed regularly according to an enterprise's policy. In addition, audit or risk management staff should have input at several stages in the development lifecycle to ensure that the application will log the proper transaction details to comply with best practices and regulatory requirements.

+ Implementation of Monitoring Technologies

The advent of security log monitoring, coupled with the emergence of fraud detection software and intrusion detection systems (IDS) as a useful replacement for human supervision, has redefined the auditing function in light of new regulatory requirements. Auditors cannot possibly manually monitor or even review the transactions many Web applications perform. Therefore, monitoring of the logs must be outsourced or completed through scripts or with a software product. An approach that integrates the expertise of the risk management or auditing team with the technical knowledge of the development staff has proven successful for organisations with sophisticated IT organisations. This collaboration is vastly more effective when it occurs before the application is put into production.

IDSs are useful in several ways. First, they have become synonymous with 'due care' in a number of interpretations of privacy legislation. Second, they provide invaluable information in coordination with security logs for reconstructing a transaction, especially one in which fraud or a computer security incident is suspected. Paired with fraud detection software, IDSs are an excellent way to monitor business processes, such as wire transfers. The management of these systems is usually outsourced to a Managed Security Services Provider (MSSP) in the case of IDS, while fraud detection software is usually employed internally. When investigating an incident or reconstituting an old transaction, IDS and fraud detection logs can be valuable resources.

Conclusion

Mitigating the risk involved in transferring the responsibility for business processes from human auditors to applications has two primary challenges: managing the vulnerabilities from third-party software (largely a matter of patching and configuration) and managing the vulnerabilities from the software an enterprise develops itself. The latter step involves both procedural and technical controls, which should serve two purposes: ensuring the confidentiality, integrity and availability of the business process once it is in production and reducing the cost of security incidents. In addition, public companies or any enterprise doing business with California customers can achieve compliance with applicable new regulations through procedural and technical diligence in application design.

Positioning application security as a regulatory requirement, business enabler and liability limiter, depending on the audience, is the ideal way to take the proper measures in safeguarding applications and the business processes they support. Whether an application succeeds or fails while executing a business process, it is imperative that the enterprise knows whether the application or compensatory systems will react the right way, ultimately guarding the bottom line.

+ For More Information

For more information about VeriSign Managed Security Services, please call 0800 032 2101 or email sales@verisign.co.uk.

Visit www.verisign.co.uk for more information.

Note: In February 2004, VeriSign, Inc., acquired Guardent, a recognised leader¹ in managed security services. Guardent's security-consulting and managed services are integrated into VeriSign's solution portfolio.

¹See www.gartner.com/reprints/guardent/118599.html for more information.

© 2005 VeriSign UK Ltd. All rights reserved. VeriSign, the VeriSign logo, and other trademarks, service marks, and designs are registered or unregistered trademarks of VeriSign and its subsidiaries in the United States and in foreign countries.

00018926