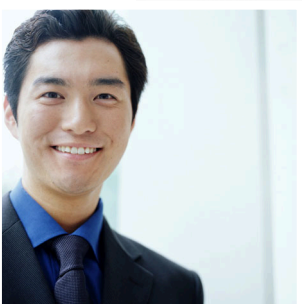


## Managing Application Security in Business Processes



## CONTENTS

Introduction	3
The Issue	3
Procedural Strategies – The Development Lifecycle	4
Vulnerability Identification	5
SEPARATE DEVELOPMENT AND QUALITY ASSURANCE ENVIRONMENTS	6
PATCH DISTRIBUTION	6
TEMPORARY COUNTERMEASURES	6
FAILURE MODES	7
CHANGE CONTROL AND MANAGEMENT	7
ADDITIONAL MEASURES – POLICY AND DEVELOPER AWARENESS	8
Technical Strategies	9
ASSESSING WEB APPLICATION VULNERABILITIES	9
IMPLEMENTATION OF LOGGING	11
IMPLEMENTATION OF MONITORING TECHNOLOGIES	11
Conclusion	12

## Introduction

As business process automation started to take hold in the early 1990s, organizations 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, organizations continued to use human intervention, believing that the supervision of auditors was required to ensure accurate money transfers and appropriate access controls to sensitive information.

Today, there is no division of labor 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 upon which they depend. However, many of these applications, especially those that are Web-based applications, 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, they at the same time expose organizations to a considerable amount of security risk versus 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 that those vulnerabilities pose to business processes.

## The Issue

“Business process” is a potentially wide-ranging term. 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 brings consequences that have yet to be fully appreciated. Organizations that value time-to-market over security often automate business processes using Web applications. In many cases, these organizations 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, human auditors were relied upon exclusively 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 Audit-Ability** — What information does the business process record about itself and can the business process be reconstructed after the fact?

The Sarbanes–Oxley Act, a recent legislation that affects every publicly traded company, mandates that audit-ability controls must be in place for financial transactions. In addition, California has passed a Notice of Security Breach, CA Civil Code 1798.82<sup>1</sup> (also known as SB1386) that mandates notification if there is a confirmed or suspected compromise of sensitive customer information.

The information systems that are 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 business process protection strategy will include a well-described inventory system for critical business processes, diligent monitoring programs 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.

<sup>1</sup> Other sections of the law include CA Civil Code 1798.29 and 1798.84. Act also referred to as the California Database Protection Act or the California Security Breach Information Act.

- Planning for failure modes, ensuring that damage is minimized 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 of 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.

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	Post-Implementation Reviews
Develop Change Control Strategy	Develop Contingency-Back Out Plans		
Secure Approvals	Create Training Materials	Secure Final Approval and Go-Live Decision	

### 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 a complex business applications, especially those developed internally. (For strategies to proactively approach these problems see Additional Measures – Policy and Developer Awareness on page 8.) Unfortunately, commercially developed applications are often equally as insecure, therefore requiring a vigilant patching process.

*1 Other sections of the law include CA Civil Code 1798.29 and 1798.84. Act also referred to as the California Database Protection Act or the California Security Breach Information Act. For more information see [http://info.sen.ca.gov/pub/01-02/bill/sen/sb\\_1351-1400/sb\\_1386\\_bill\\_20020926\\_chaptered.html](http://info.sen.ca.gov/pub/01-02/bill/sen/sb_1351-1400/sb_1386_bill_20020926_chaptered.html).*

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 since development boxes are often configured insecurely, they should be removed from the production environment. This is even more important when the production environment is a DMZ that is 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 on both an application and system level. 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 organizations 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 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 the 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 organization must build in multiple checks during the development process to proactively avoid failures. Nonetheless, the possibility of a failure cannot be completely eliminated. For an organization to effectively manage its risk potential, 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 organization should have provisions in place for the overall business process (e.g., a rerouted transaction that does not depend on the application or the means to manually perform the application's functions), but most importantly, an application should never "fail open." Failing open 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 that are 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 computer security incident response efforts if thorough change control and 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, organizations 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 the manipulation and storage of data. 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. The deadline for Section 404 compliance was September of 2003.

**ADDITIONAL MEASURES – POLICY AND DEVELOPER AWARENESS**

It is essential to properly educate developers 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 high-level enough to meet the needs of the full breadth of development projects in the enterprise. However, note that developers will often disagree about the best method to achieve security in any given piece of code and flexible guidelines may be developed for more specific guidance. Peer review is also a useful part of the Development Lifecycle. A developer who is writing code that will impact a business process should possess a basic knowledge of the topic, which is outlined in the following diagram:

Introduction to Information Security and Privacy	Secure Web Application Development	Secure Configuration Management	Final Presentation
<p style="text-align: center;">→</p> <p>Overview Information Security and Privacy Concerns</p> <p>Overview Common Attack Techniques and Tools</p> <p>Benefits of Effective Controls</p> <p>Sources for Additional Information</p>	<p style="text-align: center;">→</p> <p>Secure Application Development</p> <p>Secure Database Design</p> <p>Operating System and Platform Considerations</p> <p>Network Security Considerations</p>	<p style="text-align: center;">→</p> <p>System Development Lifecycle</p> <p>Secure Baseline Configurations</p> <p>Change Control and Management</p> <p>Security Testing in QA</p>	<p>Instructor Presentation</p> <p>Student Presentation</p> <p>In-Class Example</p>

The Technical Strategies section will provide additional detail on the first two stages: Introduction to Information Security and Privacy and Secure Web Application Development.

## Technical Strategies

For applications that are already in place, there are a variety of technical strategies to manage vulnerabilities. The first is an assessment methodology to eliminate Web application vulnerabilities. The second is to ensure that logged applications are compliant with relevant regulatory requirements. The final piece is the logical implementation of monitoring technologies.

### ASSESSING WEB APPLICATION VULNERABILITIES

There are three basic ways to assess Web application vulnerabilities: via manual testing by a security consulting firm, via a software product conducting automated testing or via manual testing through internal personnel. For most companies, including large organizations, it is not cost-effective to internally cultivate the highly specialized skills needed to perform Web application vulnerability assessments. The former 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 trained specialized personnel to review Web applications from a black box or white box (code review) perspective and the consultants were able to capitalize on their experience with the developers' common mistakes. This eliminates a large proportion of security flaws, as shown in the following table, Top Ten Vulnerabilities in Web Applications. 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 concerns such as regulatory requirements (often regarding encryption) and service level agreements with third parties. This is still the preferred approach for enterprises that cannot abide security issues with their applications, such as financial institutions or manufacturing firms.

#### Automated Testing

Automated testing has matured in the last two 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.

<b>TOP VULNERABILITIES IN WEB APPLICATIONS</b>		
<b>A1</b>	<b>Unvalidated Parameters</b>	Information from web requests is not validated before being used by a web application. Attackers can use these flaws to attack backend components through a web application.
<b>A2</b>	<b>Broken Access Control</b>	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 unauthorized functions.
<b>A3</b>	<b>Broken Account and Session Management</b>	Account credentials and session tokens are not properly protected. Attackers that can compromise passwords, keys, session cookies, or other tokens can defeat authentication restrictions and assume other users' identities.
<b>A4</b>	<b>Cross-Site Scripting (XSS) Flaws</b>	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.
<b>A5</b>	<b>Buffer Overflows</b>	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.
<b>A6</b>	<b>Command Injection Flaws</b>	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.
<b>A7</b>	<b>Error Handling Problems</b>	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.
<b>A8</b>	<b>Insecure Use of Cryptography</b>	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.
<b>A9</b>	<b>Remote Administration Flaws</b>	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.
<b>A10</b>	<b>Web and Application Server Misconfiguration</b>	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.

### IMPLEMENTATION OF LOGGING

Regulatory requirements and the maturation of correlation technologies have driven organizations to devote the appropriate resources to properly implement logging on all systems and supporting business processes. Standard implementations of syslogs have created a baseline for gathering normalized data, but processes for reviewing and correlating the logged information have lagged slightly behind. Today, Managed Security Service Providers (MSSP) offer correlation services, security software vendors sell correlation engines and logging is required of financial transactions by law for all public companies, per Sarbanes-Oxley. 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 IDSs 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 that many Web applications perform. Therefore monitoring of the logs must be outsourced, completed through scripts or via 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 organizations with sophisticated IT organizations. It should be noted that 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. Secondly, 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 an 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 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, for public companies or any enterprise doing business with California customers, compliance with the applicable new regulations can be achieved 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.

## About VeriSign

VeriSign Inc. (NASDAQ:VRSN) delivers critical infrastructure services that make the Internet and telecommunications networks more intelligent, reliable and secure. Every day VeriSign helps thousands of businesses and millions of consumers connect, communicate, and transact with confidence. For more information on VeriSign's full line of security services, visit: **[www.verisign.com](http://www.verisign.com)**.

**Note:** In February 2004, VeriSign acquired Guardent, a recognized leader<sup>2</sup> in Managed Security Services. Guardent's security consulting and managed services are integrated into VeriSign's solution portfolio.

<sup>2</sup> See <http://www.gartner.com/reprints/guardent/118599.html> for more information.