Building web application security into your development process: are your web applications vulnerable?

White paper

Table of contents

Introduction

Information security is constantly evolving. Companies realized that electronic information is a corporate asset and should be protected, and they started protecting their information by using firewalls for network protection. Security progressed with intrusion detection systems (IDS) to monitor network traffic. Information security now addresses web applications; however, many companies do not know that their corporate assets may be exposed even with firewalls and IDS. This exposure results when web applications are not developed with security in mind.

About 70 percent of today's security breaches result from vulnerabilities in web applications. For example, the New York Times and eBay have both publicized security breaches on their websites. In both cases, attackers used the web applications to access confidential information. From the New York Times website, hackers stole confidential donor information, including Social Security numbers and donation amounts. From eBay, hackers stole user names and passwords. They then stole customer credit card information by setting up fraudulent auction sites that mimicked eBay. These two, well-publicized events demonstrate the power of application-level attacks and confirm how web applications are vulnerable even with proper firewall and IDS protection.

To prevent similar problems, you need to consider security, not only from an operations perspective, but as an integral part of the entire development lifecycle, starting when you develop your web applications. You should also use structured development processes. Strong, repeatable development processes produce better quality code in less time than unstructured processes. They also result in efficiency and effectiveness for your organization.

A structured, repeatable development process produces a better application only if you systematically include all aspects of application development. Development processes that ignore any aspect of application development can produce vulnerable applications with poorly architected applications.

Many developers overlook security when they develop web applications for three reasons.

First, web application security is relatively new. Traditionally, organizations have focused on networks and servers for security. However, even architectures with secure networks and servers can be attacked if applications are not secure.

Secondly, development tools such as Microsoft® Visual Studio®, BEA WebLogic and Macromedia ColdFusion provide powerful development environments for improving development productivity. However, these environments require careful configuration so that only the appropriate services run in production. Unfortunately, many development groups do not configure their production environments properly, leaving their web applications vulnerable.

Finally, developing secure web applications is usually an afterthought. Because security is typically not included in functional requirements, users do not focus on it, and developers do not build security into the applications. Some developers may not think that application security is necessary. They may think that if they use Microsoft Internet Information Services (IIS) or run their applications behind a firewall, the applications are secure from malicious attacks. Even developers who recognize the importance of web application security usually see it as part of the quality assurance (QA) process. As a result, many web applications can be rich in functionality but vulnerable to unwanted intrusions and attacks.

Furthermore, many development organizations view security as a one-time activity during the development process. In these cases, security becomes the responsibility of one group within the organization, such as the QA team or internal audit department. Once the group signs off on an application, the organization considers it secure.

However, web applications are not static systems. Changes to web applications create risk, and what was once secure can now be vulnerable. If security is a one-time activity, a vulnerability that enters the system after the audit can go undetected. Instead, you need to view application security as a process, included throughout the development lifecycle in order to create secure web applications. Add security into the practices of every team member associated with developing and running your web applications.

This white paper describes how to add web application security development practices to a typical object-oriented development process. Examples use many common models within the Unified Modeling Language (UML). However, these practices are generic, and you can add them into any structured development process.

Figure 1: Common development lifecycle methodology

Security as a process

Integrating security throughout the development lifecycle is a paradigm shift for many development groups. While many development teams claim that they view security as a process, in reality they do not give security the focus it requires. For example, many organizations limit security needs to the technical requirements section of a requirements document. The description is usually broad, such as "Develop the product using IIS." This level of detail does not provide developers with the guidance they need to develop applications securely. As a result, security requirements do not have high priority, and they get passed to the operations department to complete as part of deployment.

You need to address security throughout the development lifecycle. This includes defining security as part of the functional and technical requirements for an application. Once you complete the requirements, you should model security as part of the analysis and design of the application. The development team should use secure coding practices. The QA team should build and execute its test plan to address security, and you should deploy the application in an environment that has been hardened for security. Once deployed, conduct security audits in the production environment periodically to help the application remain secure when updated. This is the process of web application security.

The remaining sections of this paper describe the phases of the application development lifecycle and how to add web application security into each.

Development lifecycle

The development lifecycle begins with common terminology and processes. You can abstract all development methodologies into the following phases: requirements gathering, analysis and design, development, quality assurance, deployment and post deployment. Figure 1 shows this methodology.

Whether your development process uses extreme programming (XP), iterative, waterfall or some other derivative, each methodology contains tasks that you can map to these phases. While different methodologies may use different names for the phases and the number of phases may vary, the type of work is the same. This paper uses these general phases to describe how to add security into your development process.

Figure 2: Use case model of an online checkout system

Defining secure requirements

Creating secure web applications begins in the requirements gathering phase. To gather requirements, many organizations develop use cases that describe the functional process of a system. Use cases are usually developed with functional experts to describe how the system will work. Use cases describe proposed business processes and the steps the system will perform to complete those processes. Use cases also describe how external entities, called actors, will interact with the system. When you develop use cases, you should consider security as a business process. Each use case should contain information that answers the question, "What process makes my corporate assets and customer information secure?" You may include steps within a use case or abstract security requirements into a separate use case for clarification and reuse.

For example, a web application lets customers purchase products using the Internet. The system contains one primary use case called Purchase Product. As part of the purchasing process, the application needs to authenticate a user to determine that the user is a valid customer. Authentication is a form of application security that you can model as a process. Authentication can be as simple as validating a user name and password or as complex as validation through multiple levels of security, including timed-entry processes. The requirements must adequately describe the detail required for this process. Figure 2 shows the simple use case for a customer purchasing a product.

The use case should also explain what the system should do if the user is not authenticated. Modeling application security using use cases can educate end users on the importance of security within the application. It can also provide specific direction for developers, eliminating assumptions they may have.

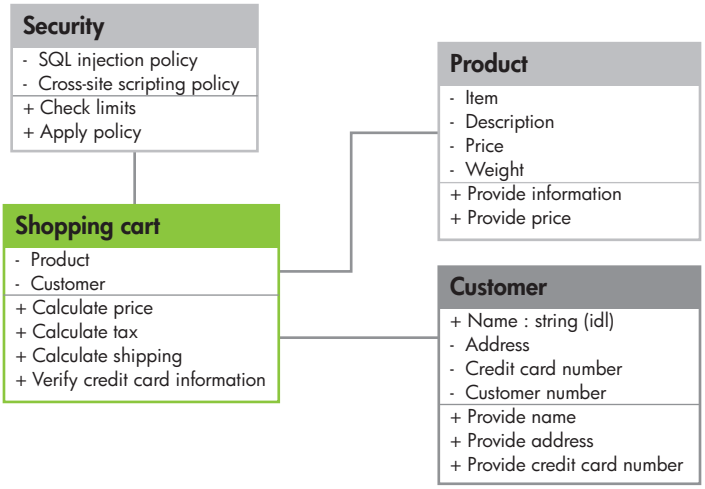
In addition to modeling security processes, you should establish business rules that further define application security. Business rules are functional aspects of the system that you cannot model as a process. For example, you can create a business rule that passwords expire after an account is inactive for six months. You can document business rules in a use case or independently, depending on how your development group defines requirements. Either way, set practices for managing business rules as part of your overall requirements gathering process.

Do not limit your application security requirements to functional requirements. Include them with technical requirements as well. Technical requirements define the features that are not user-defined, including server configuration, capacity, performance, scalability and security. Technical requirements that define application security should focus on how to protect an application from intrusive forms of attacks. Example requirements include:

- Time out sessions after 10 minutes of inactivity so that a user has to log back into the application.
- Validate all incoming data for proper format before processing it.
- Only leave Port 80 open on production web servers.

Documenting technical security requirements helps include security in later phases of the development lifecycle.

Figure 3: Security classes



Addressing application security in analysis and design

Web application analysis and design include developing class diagrams and interaction diagrams. While you can build other diagrams, such as state transition and component diagrams, during this phase, the class diagram and interaction diagram are the primary models. Analysis and design models describe the logical and physical interactions among the objects of the system. The class diagram depicts all the objects of the system and describes their interactions. Interaction diagrams are commonly called sequence diagrams and depict how objects interact to complete a specific function. The functions are usually use cases, but they can also be infrastructure and foundation interactions. You should address security within each of these models. Ideally, development groups should address security as independent objects within the model.

Continuing the online checkout system example, you can develop security into the class diagram as shown in Figure 3.

In this model, the customer and the product interact with the shopping cart to complete a transaction. Without security, the model is incomplete. However, security objects play an important role. The security objects contain the attributes and methods for preventing malicious attacks. You should explicitly model security to address appropriate security questions during this phase of the project.

Another important analysis model is the interaction diagram. This model describes how objects interact to complete the process flow within a use case. It links the functional use case and the objects. You can model security in several different ways, depending on how it is modeled in the use case and class diagram. If security is modeled as a use case, create an interaction diagram to model it. Security objects are modeled as object lifelines within each diagram that requires them. Within the use cases, individual steps that describe security are modeled as line items within the interaction diagram.

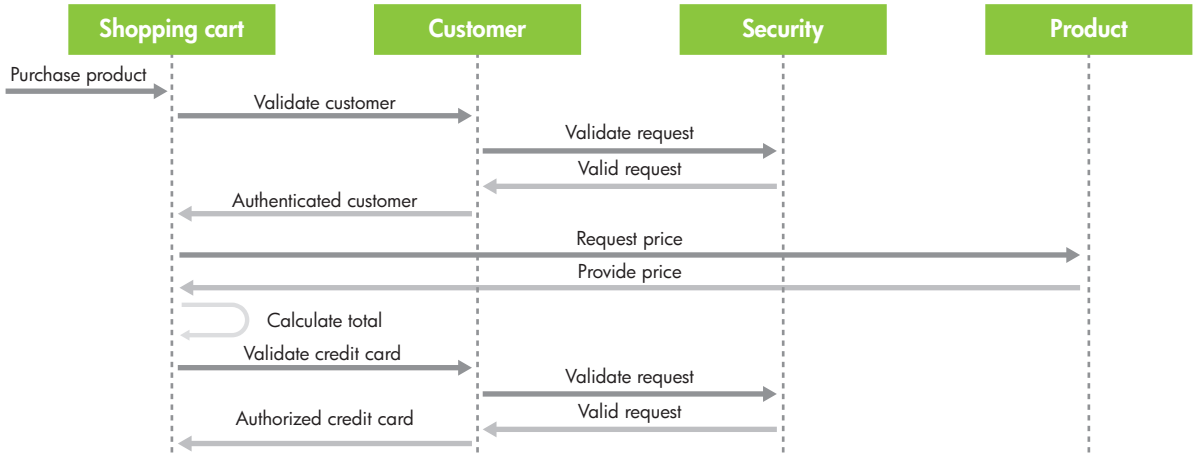
Continuing the online checkout example, the interaction diagram, shown in Figure 4, displays how the objects interact to complete the checkout process.

In this example, security is modeled both as an object lifeline and a step within the process. The security object lifeline performs the security checks modeled in the class diagram, while the steps are the same steps modeled within the use case diagram. Once you diagram these models, you can begin coding.

You should expand your current development process to include web application security. Your existing development tools should support the additional processes needed for security requirements, perform the analysis and design security modeling. However, the remaining development phases require additional capabilities for secure applications.

Web application assessment products, such as HP WebInspect software, can support these phases. HP WebInspect lets you simulate advanced web application attacks. Developers, QA testers and operations administrators can use HP WebInspect to identify specific web application vulnerabilities. The results can be proactively cycled back through the development process to correct vulnerabilities and mitigate risks. The following sections describe how to construct and implement secure web applications, using products such as HP WebInspect for web application protection.

Figure 4: Online checkout security interaction



Coding for web application security

Security requirements and models provide direction for developing application code. However, if an application is not built securely, the requirements and the models don't matter. Building secure applications requires developing and enforcing secure coding practices. Secure coding practices include not just the application code but other aspects such as user interface (UI) development and prototyping.

Secure coding practices

Most organizations have some coding standards, such as naming conventions and documentation requirements. Adding security to standard coding practices can provide immediate payback in producing more secure applications. Examples include:

- Never trust incoming data, and always check this data.
- Never rely on the client to store sensitive data.
- Make error messages generic to users, and document them for support purposes.
- Use object inheritance, encapsulation and polymorphism wherever possible.
- Take care when you use environment variables, and always check boundaries and buffers.

Once you establish secure coding practices, you must enforce them. You can use a code walkthrough, where one developer, preferably a senior developer, reviews code written by another developer. The reviewer audits the code for security to see that the code conforms to the organization's coding standards. Code that does not conform to the security standards must be corrected before it goes into the testing cycle.

Organizations that follow iterative development or XP methodologies can quickly follow these practices. Developers are teamed in pairs to develop a function. Each developer audits and reviews their partner's code from a security perspective. This practice tightens security procedures and reduces the vulnerabilities that pass into the testing process.

Designing secure user interfaces

You should also be careful about the type of data that users can access through the user interface design. Build limits into your applications to restrict the data that users can enter into form fields. In addition, restrict fields as to the type of information that users can enter. For example, limit domestic ZIP codes to numeric values only.

Once you design the interface, developers should have users test the interface from a security perspective, focusing on what the user should and should not be able to do. Compare the results with the requirements, analysis and design models and code to determine whether your applications support appropriate user actions and prevent unauthorized actions. This iterative process helps you identify and correct vulnerabilities before you release applications into production.

Prototyping security into your application

Prototyping is key to iterative web development. Through prototyping, you can break down complex modules and conduct proofs of concept for specific functionality. Unfortunately, prototypes often go into production and are difficult to support. Prototypes are not intended for production environments. They usually do not have the infrastructure of a production system. For example, security is usually not in a prototype. You should not release prototypes into production unless you enhance the prototype to include a minimum set of infrastructure requirements, including adequate application security. In addition, you should audit all production applications for security vulnerabilities so that you can identify security problems quickly.

Testing for security

Testing begins early in the software development lifecycle and extends until deployment. QA testers check that an application works according to its requirements. However, even in mature development organizations, some requirements may be implied or undocumented. Because security, in particular, is often not considered, testers often provide the only protection between a potential security problem and the assets of their organization, and their security testing must be thorough.

The first step in security testing is to include security in the test plan. Security should have its own section so that it is adequately addressed during the QA process. While traditional system testing checks that an application performs its required functions, security testing checks that the application does not perform certain functions. You should also develop security scenarios that define potential malicious attacks, such as brute-force hacking, parameter checking, SQL injection, buffer overflows and cross-site scripting. For each scenario, create an attack that a malicious intruder may use to compromise the application. Once the test plan is complete, define test cases to test the methods that attackers use to break applications. The test cases should describe the action that is tested, the expected results and whether the test passed. You must complete security testing during the QA phase before applications move into production.

Deploying secure applications

Even after an application has passed QA testing, it may be vulnerable to malicious attacks if the application is deployed in an insecure environment. Therefore, apply security checks to your production environment to determine that production servers are secure, including:

- Develop a backup and recovery plan.
- Limit access to the server, and regularly change administrator passwords.
- Close all web server ports except those needed for the application to run on the server.
- Define and secure the network connections with other servers and other networks.
- Apply operating system patches to servers on a regular basis.
- Do not leave extraneous files with old file extensions on any web server that is available to the public.
- Identify only essential services for the web server, and eliminate others.

You should develop and maintain these procedures for a production environment and then periodically audit them to verify that they are being followed.

Security in a post-implementation environment

Web applications are not static. Content alters, and new features are added, in some instances continually. Each time a web application changes, you risk application security. Even small changes can produce a vulnerability that poses a major threat to the assets of a company or provides information about a company's customers. By including proper security support in all aspects of the development process and remaining vigilant about security, you can reduce your risk of deploying applications with security vulnerabilities and avoid delays in deployment.

Yet security vulnerabilities can appear in even the most secure organizations. To mitigate this risk, an independent company should conduct penetration testing on the web application server farm periodically.

About HP WebInspect

HP WebInspect helps you secure your entire network with intuitive, intelligent and accurate processes that dynamically scan standard and proprietary web applications for known and unidentified application vulnerabilities. HP WebInspect provides a new level of protection for your critical business information. With HP WebInspect, you can find and correct vulnerabilities at their source before attackers can exploit them.

Whether you are an application developer, security auditor, QA professional or security consultant, HP WebInspect provides the capabilities you need for verifying the security of your web applications. HP WebInspect addresses the complexity of Web 2.0 and new web technologies such as Ajax, and identifies vulnerabilities that are undetectable by traditional scanners. HP WebInspect tackles today's most complex web application technologies with breakthrough testing innovations, including simultaneous crawl and audit (SCA) and concurrent application scanning, resulting in faster and more accurate automated web application security testing. HP WebInspect lets you perform security assessments for any web application, including these industry-leading application platforms:

- Macromedia ColdFusion
- Lotus Domino
- Oracle® Application Server
- Macromedia JRun
- BEA WebLogic
- Jakarta Tomcat
- ASP and ASP.NET

Conclusion

Your web applications are a portal to your corporate assets. You need to use the necessary security procedures to protect those assets from malicious attacks. Security has evolved from networks and servers and now includes applications, and you need to address all three. Incomplete development processes leave your applications at risk, no matter how structured your development process may be. For greater application security, you need to use mature development practices that focus on web application security.

You can add secure practices into your development process. Implementing a few security methodologies into an existing methodology provides immediate results and better quality applications.

Business case for application security

Whether a security breach is made public or confined internally, the fact that a hacker can access your sensitive data is a huge concern to your company, your shareholders and most importantly, your customers. Companies that are vigilant and proactive in their approach to application security are better protected. In the long run, these companies enjoy a higher return on investment for their e-business ventures.

To learn more, visit www.hp.com/go/software

4AA1-5409ENW, October 2007

invent