Systems Development Advice in a Web-enabled World

By Sanjiv Kumar Agarwala, CISA, CISSP

Real estate, shopping malls, event management, airlines, hospitals, stocks and trade—practically every business, small or large, has its own web site, and customers make online purchases by a mere mouse click. It is truly an anywhere, anytime business model. On the other side of the spectrum, there are also business applications for financial management, inventory management and other routine, in-house business processes. These applications are mission-critical, as they are directly linked to the organization's goals and mission. A business application takes customer information and trading parameters as input; processes this information based on the business logic embedded within the application; and gives timely, reliable and continuous information as output to drive the business.

Organizations are highly dependent on the running of their business applications. Loss of sensitive customer information, business data and trade secrets; unavailability of the services for a few hours; or tampering data in business systems would adversely affect business. Enterprises are continuously moving toward a phenomenon known as web-enabled applications, and with the advent of web services and XML standards for interoperability, this becomes even more obvious. More and more business applications use web application development, as it takes less time for development and is accessible to a huge user base. Even some legacy applications have become web-enabled by introducing web application as an interface.

Two major critical situations can arise: fraud and application attacks. In a fraud, the application can be used inappropriately to make a financial transaction that satisfies a business equation, but is not actually allowed by business. Recent frauds at major organizations are examples of this kind of attack. In application attacks, a business application is attacked to make it behave abnormally and/or to steal information. Such attacks include application denial of service, stealing confidential information and tampering with information in transit.

This article discusses the traditional system development life cycle (SDLC) approach in developing an application, the common web application attacks, the perspective of how an attacker thinks like a developer in carrying out an attack and what improvements can be made in the traditional system development life cycle approach. Relevant *Control Objectives for Information and related Techology* (COBIT) controls are also cited as a guide to developers.

Application Development Process

The traditional systems development life cycle approach includes:

- Feasibility—Determination of the strategic benefits of implementing the system either in productivity gains or in future costs avoidance, identification and quantification of the cost savings of a new system, and estimation of a payback schedule for costs incurred. This business case provides the justification for proceeding to the next phase.
- Requirement definition—Defininition of the problem or need that requires resolution, and of the functional and quality requirements of the solution system. This can be either a customized approach or a vendor-supplied software package, which would require following a defined and documented acquisition process. In either case, the user needs to be actively involved.
- Design—Based on the requirements defined, establishment of a baseline of system and subsystem specifications that describe the parts of the system; how they interact; and how the system will be implemented using the chosen hardware, software and network facilities. Generally, the design also includes program and database specifications, and a security plan. Additionally, a formal change control process is established to prevent uncontrolled entry of new requirements into the development process.
- Development—Use of the design specifications to begin programming and formalizing the supporting operational processes of the system. Various levels of testing also occur in this phase to verify and validate what has been developed.
- Implementation—Establishment of the actual operation of the new information system, with final user acceptance testing conducted in this environment. The system may also go through a certification and accreditation process to assess the effectiveness of the business application in mitigating risks to an appropriate level and providing management accountability over the effectiveness of the system in meeting its intended objectives and in establishing an appropriate level of internal control.
- Post-implementation—Following the successful implementation of a new or extensively modified system, a formal process that assesses the adequacy of the system and projected cost-benefit or return on investment (ROI) measurements. In so doing, information system (IS) project and end-user management can provide lessons learned and/or plans for addressing system deficiencies, as well as recommendations for the future projects regarding system development and the project management processes followed.

Attacks on Applications

The 10 most common web application attacks are:

- Unvalidated input—Programs take input from users through form fields, from parameters passed in URLs and hidden fields, or from cookies. There are a number of ways the information can be encoded and sent as input to the application, and it is quite possible that some of the validations (checks for specific legal values and input patterns) for the input are missing, and are sent to the application.
- Broken access control—A user has access to the contents of an application based on the user's role and what he/she is allowed to do. During application development, there could be an interface that gives unauthorized users access to data not meant for them.

- Broken authentication and session management—The purpose of user authentication (mostly through user ID and password) is to allow the right users access, and session management is used to track the active sessions for those users. Flawed implementations of some interfaces (such as password change, editing account information and passing session ID in cleartext over the wire) break the authentication and session management as unauthorized users can act like an authorized users.
- Cross-site scripting flaws—This occurs when a user is sent malicious code in the form of a script and his/her browser executes it without validating it, thinking it is from a trusted source. Sensitive information can be stolen from the user's local system by exploiting this flaw.
- Buffer overflows—Attackers give specially crafted input to the input fields (especially strings and arrays), which have no bound checks, thus overflowing the stack and transferring control to the malicious code. The malicious code may result in giving the shell/command prompt to the attacker.
- Injection flaws—Many applications use operating system features and external programs to perform their function, like accessing a back-end database through SQL or sending mail through sendmail. An attacker can embed a malicious SQL command in the input field. This input, if not validated before being sent to the database server for execution, might give additional information or help in destructive activities, such as dropping a table.
- Improper error handling—An application gives an error when it does not know how to handle an exception. Some messages like "IIS web log full" or "unhandled exception occurred in function x( ) in DLL myApp" give away a lot of information, which helps the attacker to design attacks by exploiting these problems.
- Insecure storage—Sensitive information may be stored in files, cookies or in databases in plaintext, or it may be stored with a weak cryptographic algorithm after it is encrypted. The attacker may access these storage systems and retrieve the passwords and other sensitive information easily.
- Denial of service—An attacker can send a large number of requests to the web applications, which eat away resources, making the service unavailable to authorized users. The system may become so slow it needs to be started again.
- Insecure configuration management—An application might be deployed with a default user ID and password (e.g., user admin with password admin), or it may be using the buggy, old version of the software. This gives the attacker easy access to exploit the application.

Food for Thought

Developer

An application developer is working on a user management module of a large project.

- He defines a field "user ID." The user can be a guest, registered user or an administrator.
- He provides a "view user details page" and the page has a "comments" field where the user can log comments.
- He stores the user authentication information in the cookies.
- He sends all this information over an HTTPS connection established between the user browser and the web server.

The developer is happy that he has delivered the solution quickly and the project manager will be happy.

Attacker:

- The attacker logs into the web site as a guest and sees that there is a "members only" section that is disabled for guests.
- He sees that a role ID of 5 is passed to the server in a hidden field.
- He does a "view source" on the user details page and finds some commented code carrying a test user ID.
- He downloads the page, changes the role ID to 1 and submits the page again. He can see the "members only" section now enabled.
- He is also able to pass large amounts of data to the web application in the "comments" field, and he knows he could send the application crashing.

He signs out from the web site and tries to sign into the site again with the user ID that he discovered in the user details page. He keeps the password same as user ID and it succeeds.

The developer has to anticipate disastrous situations like an attacker passing certain input that makes the application hang or go into some exceptional condition that may cause an application crash or divulge sensitive information. Suitable care has to be taken to overcome such an eventuality.

Filling the Gaps in the Software Development Life Cycle Phases

Feasibility

- Can the solution be deployed in the proposed setup with adequate security (reasonable, to say the best), so the company can benefit strategically?

Requirement Definition

- What is the business-critical information the application is trying to process, and how will it be used?
- It has to be understood that security is an enabler in a business application. The application designer and developer have to understand that security should not become a bottleneck, but rather support the business functionality of the application.

Design

- Identify the input, processing and output controls. Identify the input fields, and judge their sensitivity. Draw a flow diagram detailing how it is used.
- Research and analyze the target environment in which the application is going to be deployed. Consider the vulnerability level of the platform and the cost of containing the vulnerabilities when deciding the platform for use.
- Check that sensitive information is not unnecessarily stored by the system when it is not going to be used/processed (e.g., Social Security number).
- Develop a security testing test plan so adequate testing can be done within the limited time.
- Maintain an audit trail of the application events or transactions, so fraudulent and information security-relevant events can be detected. Manual and automatic checking with this audit trail information helps restore the application data and make them consistent and, more important, acts as a detective control for evidence reporting.
- Perform a security audit of the application design to help find any design flaws and correct them before the actual code is written. This also saves time and money and prevents unnecessary rework.

Development

- Code analyzers like Flawfinder, RATS, etc., that flag the trouble code pieces in an application, should check if the application has defensive techniques against common web application attacks (like parameterized queries, buffer checks, string parsing, validating user and nonuser input, etc.).
- Some integrated development environments (IDE) provide the means to write secure code (e.g., .NET platform, with code access security features supported in various languages). It is equally important to understand that improper use of security functionality provided by a platform can be more problematic, which some developers fail to understand.
- Final implementation should not release binaries with debug information, as they can be reverse-engineered. If it is a UNIX-based application, strip utility should be used for creating similar binaries.
- The code should be tested and attacked from an attacker's perspective. Test cases should cover improper input, multiuser requests, network outages, corrupt or missing files, and application crashes. Applications should ensure that they do not give too much information in error message(s) and help messages to the user.
- Testing should also test for problems like cross-site scripting, SQL injection attacks, buffer overflow problems and application DoS. Developers can write their own tools or test scripts, or they can use one of many tools for testing and fixing these problems, as mentioned in **figure 1**.

Figure 1—Tools For Use During Testing

Tool Category and Description	Example Tools
Reverse engineering tools (tools for digging application dependencies, intermediate source code, stored variables, like passwords and functions, implemented within the binaries)	Dumpbin
Web page spying and analysis tools (tools for spying on and manipulating raw data, such as HTML, HTTP headers, session IDs, and cookies that are being sent between a client and server)	Netcat, cookiesspy
Sniffing tools (tools used for sniffing the data being sent over the wire between interacting servers like application server, database server and web server)	Ethereal
Password cracking tools (tools used for cracking passwords)	JohnTheRipper, Brutus, etc.
Automated web security test tools (used for automated web application testing from a hacker's perspective)	WebInspect, AppScan, Whisker

- Third-party components should not be assumed as safe. It is a responsibility to test that the application interacts in a secure manner with all third-party components, such as middle-tier components and custom controls, like ActiveX controls. All of these should be taken care of in a test plan, and it is always best to develop the test plan as early as possible in the SDLC phases.

Implementation

- It is important to secure the application during deployment. The integrity of the software should not be compromised in any way. Strong-name signing can be used as a means of guaranteeing the integrity of software.
- A digital certificate should be assigned to the application that serves as proof of identity. An X.509 certificate should be embedded into the application binaries—.EXE and .DLL files—along with an encrypted hash digest value of the application that uniquely identifies the application.
- Obfuscating the code before it is released should be considered to help make it more difficult for others to reverse-engineer the application. The obfuscation process changes the names of the functions and variables, reorganizes the code and masks the constants to make it extremely difficult for reverse engineering.
- Equally important is creating a release build of the application, not a debug build. The debug information within the debug build may carry sensitive information, and it also may be easy to reverse-engineer such an application.
- The system may also go through a certification and accreditation process to assess the effectiveness of the business application in mitigating risks to an appropriate level, providing management accountability over the effectiveness of the system in meeting its intended objectives and in establishing an appropriate level of internal control. Common Criteria for Information Technology Security Evaluation (CCITSE) can be used based on the business need and suitability of the product.

Post-implementation

- Lessons learned, best practices and recommendations should be documented and put in a central place, so that over time, other users can start gaining from the knowledge shared.
- Information like application risk assessment and ROI results, and number of potential vulnerabilities discovered and their impact can help in designing better applications in the future.

CobiT Controls for Developers

Figure 2 outlines security-specific CobiT controls. The controls mentioned here are those for which security is part of the control. There are other controls that are part of the SDLC process, and readers can refer to them in CobiT.

Conclusion

Security is a process and should become an important component during the various phases in the life cycle of system development. Implemented in the right spirit, the result is an end product that is highly reliable and takes away the burden of implementing costly security controls once the testing is over. Application designers, developers and testers should write defensive code to take care of the common problems in the web application. As more and more applications are moving toward a web-based architecture, there should be a paradigm shift in the thinking process of application development.

While the application requirements and development environment have become complex, software development practices largely have not kept up. Since attackers or hackers often successfully use the developer's mode of thinking while planning an attack, the developer should think like an attacker when he/she develops an application.

References

CISA Review Manual, 2004, ISACA, *www.isaca.org*

COBIT 3rd Edition, *Control Objectives*, IT Governance Institute, USA, 2000, *www.itgi.org*

OWASP TopTen2004, *http://heanet.dl.sourceforge.net/sourceforge/owasp/OWASPTopTen2004.pdf*

Robinson, Ed; Michael James Bond; *Security for Microsoft Visual Basic .NET*, McGraw Hill

Sanjiv Kumar Agarwala, CISA, CISSP
is an e-security consultant at Tata Consultancy Services (TCS) with more than five years of industrial experience in software development, application architecture, penetration testing, network security assessment and audits. He has executed various projects and consultancy assignments and is involved in research activities. He is also an active member of the ISACA Pune (India) Chapter.

Figure 2—COBIT Controls for Developers

COBIT Control	Control Objective
PO10: Manage Projects 10.9 Planning of Assurance Methods	Assurance tasks are to be identified during the planning phase of the project management framework. Assurance tasks should support the accreditation of new or modified systems and should assure that internal control and security features meet the related requirements.
AI1: Identify Automated Solutions 1.9 Cost-effective Security Controls	Management should ensure that the costs and benefits of security are carefully examined in monetary and nonmonetary terms to guarantee that the costs of controls do not exceed benefits. The decision requires formal management sign-off. All security requirements should be identified at the requirements phase of a project and justified, agreed and documented as part of the overall business case for an information system. Security requirements for business continuity management should be defined to ensure that the planned activation, fallback and resumption processes are supported by the proposed solution.
AI2: Acquire and Maintain Application Software 2.12 Controllability	The organization's system development life cycle methodology should require that adequate mechanisms for assuring the internal control and security requirements be specified for each information system development or modification project. The methodology should further ensure that information systems are designed to include application controls, which guarantee the accuracy, completeness, timeliness and authorization of inputs, processing and outputs. Sensitivity assessment should be performed during initiation of system development or modification. The basic security and internal control aspects of a system to be developed or modified should be assessed along with the conceptual design of the system to integrate security concepts in the design as early as possible.
AI2: Acquire and Maintain Application Software 2.13 Availability as a Key Design Factor	The organization's system development life cycle methodology should provide for availability to be considered in the design process for new or modified information systems at the earliest possible stage. Availability should be analyzed and, if necessary, increased through maintainability and reliability improvements.
AI2: Acquire and Maintain Application Software 2.14 IT Integrity Provisions in Application Program Software	The organization should establish procedures to assure, where applicable, that application programs contain provisions that routinely verify the tasks performed by the software to help assure data integrity and provide the restoration of the integrity through rollback or other means.
AI2: Acquire and Maintain Application Software 2.15 Application Software Testing	Unit testing, application testing, integration testing, system testing, and load and stress testing should be performed according to the project test plan and established testing standards before implementation is approved by the user.Adequate measures should be conducted to prevent disclosure of sensitive information used during testing.
AI5: Install and Accredit Systems 5.10 Security Testing and Accreditation	Management should define and implement procedures to ensure that operations and user management formally accepts the test results and the level of security for the systems, along with the remaining residual risk. These procedures should reflect the agreed-upon roles and responsibilities of the end user, system development, network management and system operations personnel, taking into account segregation, supervision and control issues.

Information Systems Control Journal, formerly the IS Audit & Control Journal, is published by the *Information Systems Audit and Control Association*, Inc.. Membership in the association, a voluntary organization of persons interested in information systems (IS) auditing, control and security, entitles one to receive an annual subscription to the *Information Systems Control Journal*.

www.isaca.org