Application Developer Security Guidelines

Application developers should include information security features in the application design itself, as information should be protected from the time it is first collected until the time it is officially disposed of. As part of the design phase, specific security controls should be selected. The information resources (e.g., hardware, software) required to support the selected security controls should be identified and converted to functional specifications.

Application designers and developers should determine the applicable regulatory and legal requirements for the application, based on the types of information that will be processed by the application. For example, applications processing credit card information are in scope for Payment Card Industry (PCI) standards. Applications processing personally identifiable information (PII) are in scope for data privacy regulations, which can vary by country.

Controls

The following general controls should be considered, at a minimum:

- User functionality (including user interface services) should be partitioned from information system management functionality.
- Security functions should be isolated from non-security functions.
- Firewall restrictions to application connectivity should be considered including any potential exceptions to existing firewall rule standards.
- Code Access Security should be implemented when developing managed code.
- Privileges assigned to the application should be appropriate, e.g., applications should not be required to run under Administrator, Root or superuser accounts.
- Encryption requirements (for both data at rest and during transmission) as defined by the data classification.

The following application controls should be considered in order to support general application control requirements:

- Providing menus to control access to application system functions
- Restricting user knowledge of information and application system functions that they are not authorized to access with appropriate editing of user documentation
- Controlling the access rights of users (e.g., read, write, delete, and execute)

- Ensuring that outputs from application systems handling sensitive information contain only the information relevant to the use of the output
- Controls included that protect the application from service interruption such as dedicated hardware, redundant hardware configurations, adequate processing and storage capabilities, transaction recovery and resilient data storage (e.g., disk mirroring and RAID technology).

Threat models should be developed during the Application Design phase. The threat models should identify potential attack vectors, performance modeling, vulnerabilities inherent in the technologies used, and any other potential threats to the application or information processed within the application. Threat models should be developed in cooperation with development resources, security analysts and business representatives and documented within the Application Design documentation. Controls identified as necessary for management of identified threats should be translated into Functional Requirements.

Application Security Verification

Testing of security features and controls should be performed as part of application testing. The security features documented during application design should be thoroughly tested. Application testing should be a joint effort of users and the development teams, and should include both the manual and automated phases of the application.

Application Security Verification Tests should include, at a minimum:

- Dynamic Code Analysis: Run-time verification of software programs to ensure that a program's functionality works as designed. This verification task should specify tools that monitor application behavior for memory corruption, user privilege issues, and other critical security problems.
- Fuzz Testing: Fuzz testing is a specialized form of dynamic analysis used to induce program failure by deliberately introducing malformed or random data to an application. The fuzz testing strategy is derived from the intended use of the application and the functional and design specifications for the application. The security advisor may require additional fuzz tests or increases in the scope and duration of fuzz testing.
- Attack Surface Review: The threat models developed as part of the Design phase should be reviewed to ensure the threat models and attack surface measurement of the application are still valid and addressed when it is code complete. This review ensures that any design or implementation changes to the system have been accounted for, and that any new attack vectors created as a result of the changes have been reviewed and mitigated.

Where possible the Company should subject the software to an independent security review of the source code, especially for Internet-facing systems.

Evidence of security verification tests should be maintained for at least six (6) months and contain a classification of Confidential. Evidence should include who performed the test, the nature of the tests performed, and the results of those tests.

Verification tests should be based on the Open Web Application Security Project (OWASP) Top 10 (https://www.owasp.org/index.php/Top_10_2010-Main) and the SANS Top 25 (http://www.sans.org/top25-software-errors/), at a minimum.

Acceptance Tests

All significant modifications, major enhancements, and new systems should be acceptance tested by the appropriate users prior to installation of the software in production. The user acceptance plan will include tests of all major functions, processes, and interfacing systems. Testing procedures should be properly documented on the change request forms.

Acceptance tests should allow business users to test business functions to simulate the live environment including the complete application functionality. Additionally, acceptance testing should be conducted for security purposes as well by an assigned security representative to detect vulnerabilities, insecure use of programming features, or potential compromise the security of the system. Acceptance testing should also include appropriate stress, performance, and volume testing to simulate production usage to determine any potential failure points.

Additional Information

The SANS TOP 25 Software Errors website (http://www.sans.org/top25-software-errors/) contains resources to help eliminate coding errors related to security.

OWASP also publishes resources and additional information (https://www.owasp.org/index.php/Top_10_2010-Main).

Microsoft publishes information on their Security Development Lifecycle (SDL), including specific information and tools for the different phases of the SDL (http://www.microsoft.com/security/sdl/default.aspx).