Architecture and Design Review

How to Use This Checklist

This checklist is a companion to Chapter 4, "Design Guidelines for Secure Web Applications," and Chapter 5, "Architecture and Design Review for Security." Use it to help you perform architecture and design reviews to evaluate the security of your Web applications and to implement the design guidelines in Chapter 4.

This checklist should evolve based on the experience you gain from performing reviews. You might also want to perform custom checks that are based on a specific aspect of your architecture or design to ensure that your deployment environment the design.

Deployment and Infrastructure Considerations

Check	Description
□	The design identifies, understands, and accommodates the company security policy.
□	Restrictions imposed by infrastructure security (including available services, protocols, and firewall restrictions) are identified.
□	The design recognizes and accomodates restrictions imposed by hosting environments (including application isolation requirements).
□	The target environment code-access-security trust level is known.
□	The design identifies the deployment infrastructure requirements and the deployment configuration of the application.
□	Domain structures, remote application servers, and database servers are identified.
□	The design identifies clustering requirements.
□	The design identifies the application configuration maintenance points (such as what needs to be configured and what tools are available for an IDC admin).
□	Secure communication features provided by the platform and the application are known.
□	The design addresses Web farm considerations (including session state management, machine specific encryption keys, Secure Sockets Layer (SSL), certificate deployment issues, and roaming profiles).
□	The design identifies the certificate authority (CA) to be used by the site to support SSL.
□	The design addresses the required scalability and performance criteria.

Application Architecture and Design Considerations

Input Validation

Check	Description
□	All entry points and trust boundaries are identified by the design.
□	Input validation is applied whenever input is received from outside the current trust boundary.
□	The design assumes that user input is malicious.
□	Centralized input validation is used where appropriate.
□	The input validation strategy that the application adopted is modular and consistent.
□	The validation approach is to constrain, reject, and then sanitize input. (Looking for known, valid, and safe input is much easier than looking for known malicious or dangerous input.)
□	Data is validated for type, length, format, and range.
□	The design addresses potential canonicalization issues.
□	Input file names and file paths are avoided where possible.

□	The design addresses potential SQL injection issues.
□	The design addresses potential cross-site scripting issues.
□	The design does not rely on client-side validation.
□	The design applies defense in depth to the input validation strategy by providing input validation across tiers.
□	Output that contains input is encoded using HtmlEncode and UrltEncode.

Authentication

Check	Description
□	Application trust boundaries are identified by the design.
□	The design identifies the identities that are used to access resources across the trust boundaries.
□	The design partitions the Web site into public and restricted areas using separate folders.
□	The design identifies service account requirements.
□	The design identifies secure storage of credentials that are accepted from users.
□	The design identifies the mechanisms to protect the credentials over the wire (SSL, IPSec, encryption and so on).
□	Account management policies are taken into consideration by the design.
□	The design ensure that minimum error information is returned in the event of authentication failure.
□	The identity that is used to authenticate with the database is identified by the design.
□	If SQL authentication is used, credentials are adequately secured over the wire (SSL or IPSec) and in storage (DPAPI).
□	The design adopts a policy of using least-privileged accounts.
□	Password digests (with salt) are stored in the user store for verification.
□	Strong passwords are used.
□	Authentication tickets (cookies) are not transmitted over non-encrypted connections.

Authorization

Check	Description
□	The role design offers sufficient separation of privileges (the design considers authorization granularity).
□	Multiple gatekeepers are used for defense in depth.
□	The application's login is restricted in the database to access-specific stored procedures.
□	The application's login does not have permissions to access tables directly.
□	Access to system level resources is restricted.
□	The design identifies code access security requirements. Privileged resources and privileged operations are identified.
□	All identities that are used by the application are identified and the resources accessed by each identity are known.

Configuration Management

Check	Description
□	Administration interfaces are secured (strong authentication and authorization is used).
□	Remote administration channels are secured.

□	Configuration stores are secured.
□	Configuration secrets are not held in plain text in configuration files.
□	Administrator privileges are separated based on roles (for example, site content developer or system administrator).
□	Least-privileged process accounts and service accounts are used.

Sensitive Data

Check	Description
□	Secrets are not stored unless necessary. (Alternate methods have been explored at design time.)
□	Secrets are not stored in code.
□	Database connections, passwords, keys, or other secrets are not stored in plain text.
□	The design identifies the methodology to store secrets securely. (Appropriate algorithms and key sizes are used for encryption. It is preferable that DPAPI is used to store configuration data to avoid key management.)
□	Sensitive data is not logged in clear text by the application.
□	The design identifies protection mechanisms for sensitive data that is sent over the network.
□	Sensitive data is not stored in persistent cookies.
□	Sensitive data is not transmitted with the GET protocol.

Session Management

Check	Description
□	SSL is used to protect authentication cookies.
□	The contents of authentication cookies are encrypted.
□	Session lifetime is limited.
□	Session state is protected from unauthorized access.
□	Session identifiers are not passed in query strings.

Cryptography

Check	Description
□	Platform-level cryptography is used and it has no custom implementations.
□	The design identifies the correct cryptographic algorithm (and key size) for the application's data encryption requirements.
□	The methodology to secure the encryption keys is identified.
□	The design identifies the key recycle policy for the application.
□	Encryption keys are secured.
□	DPAPI is used where possible to avoid key management issues.
□	Keys are periodically recycled.

Parameter Manipulation

Check	Description

□	All input parameters are validated (including form fields, query strings, cookies, and HTTP headers).
□	Cookies with sensitive data are encrypted.
□	Sensitive data is not passed in query strings or form fields.
□	HTTP header information is not relied on to make security decisions.
□	View state is protected using MACs.

Exception Management

Check	Description
□	The design outlines a standardized approach to structured exception handling across the application.
□	Application exception handling minimizes the information disclosure in case of an exception.
□	The design identifies generic error messages that are returned to the client.
□	Application errors are logged to the error log.
□	Private data (for example, passwords) is not logged.

Auditing and Logging

Check	Description
□	The design identifies the level of auditing and logging necessary for the application and identifies the key parameters to be logged and audited.
□	The design considers how to flow caller identity across multiple tiers (at the operating system or application level) for auditing.
□	The design identifies the storage, security, and analysis of the application log files.