Security Review Check-list (Code)

Design

☐ architecture and design documentation is complete

☐ user and role based privileges are documented

☐ site is well partitioned into public and restricted pages

☐ security is layered - each layer assumes other layers may have been compromised

☐ security design covers all 7 principles of web security: authentication, authorization, confidentiality, message integrity, data integrity, availability, non-repudiation

☐ sensitive data has been identified

Note:

1. Authentication - Confirm something is authentic. Example: confirming the identity of a user.

2. Authorization - Specify access rights to resources. Example: only Joe can view Joe's account balance.

3. Confidentiality -Prevent the disclosure of information to unauthorized individuals or systems. Example: message encryption.

4. Data/Message Integrity –Data/Message cannot be modified or corrupted without detection.

5. Availability - Web sites need to be available and fast. Example: many websites can boast 99.99% uptime.

6. Accountability -When a person or system accesses or changes data their actions should be traceable. Example: logging

7. Non-repudiation - The ability to prove that a transaction took place. Example: electronic receipts.

Authentication and User Management

☐ user credentials are encrypted in the data store
☐ security policies are configurable (not hardcoded)
☐ standard security frameworks are used (instead of custom code)
☐ SSL is used to protect user credentials and authentication tokens
☐ authentication cookies are not persisted
☐ authentication cookies are encrypted
☐ cookie names and paths are used
☐ application handles user management events such as authentication failure, password reset, password change, account lockout and cancel account
☐ application handles suspicious events such as multiple failed logon attempts, session replay and attempted access to restricted resources
☐ strong passwords policies are enforced
☐ authentication credentials are not passed by HTTP GET

Authorization

☐ authentication and authorization should be the first logic executed for each request
☐ authorization checks are granular (page and directory level)
☐ deny access to pages and data by default
☐ re-authenticate for requests that have side-effects
☐ ACLs are configured for all files
☐ authorization based on clearly defined roles
☐ authorization works properly and cannot be circumvented by parameter manipulation
☐ authorization cannot be bypassed by cookie manipulation

Session Management

☐ no session parameters passed in URLs
☐ session cookies expire in a reasonably short time
☐ session cookies are encrypted
☐ session data is being validated
☐ private data in cookies is kept to a minimum
☐ application avoids excessive cookie use
☐ session id is complex
☐ session storage is secure
☐ application properly handles invalid session ids
☐ session limits such as inactivity timeout are enforced
☐ logout invalids the session
☐ session resources are released when session invalidated

Input/Data Validation

☐ all external input is validated without exception
☐ where possible input is restricted to known good chars
☐ data is validated server side (security should not rely on client-side validations)
☐ application validates numerical input and rejects unexpected input
☐ application efficiently evaluates input length
☐ strong separation between data and commands
☐ strong separation between data and client side scripts
☐ data should be checked for special characters before being passed to SQL, LDAP, OS and third party commands
☐ http headers are validated for each request (e.g. referrer)

Cryptography

☐ sensitive data has been secured in memory, storage and transit
☐ restricted areas require SSL
☐ sensitive information not passed to/from non-SSL pages
☐ proper SSL set up
☐ SSL provider supports only strong algorithms
☐ web based admin tools require SSL
☐ decryption services protected by authentication/authorisation
☐ require SSL for login page
☐ securely store cryptographic keys

Exception/Error Handling

☐ When exceptions occurs the application fail securely
☐ error messages do not reveal sensitive information
☐ system errors are never shown to users
☐ resources are released and transactions rolled back when there is an error

Auditing and Logging

☐ all user / system actions are logged
☐ sensitive information is not logged (e.g. passwords)
☐ logging for user management events (e.g. password reset)
☐ unusual activity such as multiple login attempts are logged
☐ logs have enough detail to reconstruct events for audit purposes
☐ logging is highly configurable (logging levels)

General

☐ Proper configuration of frameworks such as Spring, Struts, .NET etc..

☐ libraries are up-to-date

☐ system calls have their return status checked

☐ efficient memory usage

☐ no exposures to buffer overruns

☐ code, services, commands and processes are executed using minimal privileges (least privileges)

☐ code has no back doors

☐ debugging code and test harnesses have been removed