**Security Control**

The primary focus of the code review is to ensure that these security controls are in place, that they work properly, and that they are correctly invoked in all the necessary places. The checklist below can help to ensure that all the likely risks have been considered.

The design and implementation approaches used for input validation, authentication, authorization, configuration management, and the remaining areas by doing this, you create a security profile for the application.

|  |  |
| --- | --- |
| **Category** | **Considerations** |
| Input validation | Is all input data validated and DV mechanism is present?  Make sure that input is not modified by a malicious user and that attacker does not inject commands or malicious data into the application. Such as HTTP headers, input fields, hidden fields, drop down lists, and other web components are validated properly.  The proper length checks on all input exist.  Make sure that the data is valid on the server side.  Can data in the database Well-formed and contained only known good chars.  A centralized model or decentralized model is used. Where in data validation.  Assure in the validation model there are no backdoors. |
| **Authentication** | Ensure that credentials secured if they are passed over the network?  Fortify strong account policies used?  Ensure that strong passwords enforced?  Ensure you are using certificates?  Assure all password verifiers used for user passwords?  Backdoors are not present in production code |
| **Authorization** | Assure that there is authorization mechanism in place and work properly.  Ensure that What gatekeepers are used at the entry points of the application?  Ensure that authorization is checked on every request.  Assure authorization fail securely and only allow access upon successful confirmation of credentials. |
| **Cookie management** | Ensure that sensitive information is not comprised.  Ensure that proper encryption is in use.  Assure the session data is being validated.  Assure that cookies contain some private information. And entire cookies are encrypted  Identify all cookies being used by the application, their name, and why they are needed. |
| **Sensitive data** | Examine What sensitive data is handled by the application?  What type of encryption is used?  Examine how are encryption keys secured? |
| **Session management** | Examine How is session generated?  Unauthenticated and authenticated.  Examine how the application tracks sessions.  Examine How is persistent session state secured as it crosses the network?  Determine the session HTTP inactivity timeout.  Determine how multithreaded/multi-user session management is performed.  Examine how the logout functionality functions |
| **Cryptography** | Examine algorithms and cryptographic techniques are used.  Does the application put its encryption into action?  How often are keys recycled?  Ensure the application is Putting known good cryptographic methods.  Ensure No important data has been transferred internally or externally |
| **Secure code**  **Environment** | Examine all memory allocations/de-allocations.  Examine the file structure.  Examine any components that should not be directly accessible available to the user?  Assure that no development environment kit is contained in the build directories. |
| **Exception management** | Determine how the application handle error conditions.  Assure that exceptions and error conditions are properly working.  Ensure resources are released if an error occurs. And no system errors can be returned to the user.  Ensure that the application fails in a secure manner. |
| **Auditing and logging** | Determine your application audit activity across all tiers on all servers?  Examine How are log files secured?  Make sure no sensitive information is logged in the event of error.  E.g., cookies, HTTP “GET” method, authentication credentials.  Make sure that successful and unsuccessful authentication is logged, and application errors are logged |