While every codebase is unique and may have specific security considerations, here's a general checklist that can help guide your secure code review process:

Input Validation:
- Check if user input is properly validated to prevent common vulnerabilities like SQL injection, cross-site scripting (XSS), and command injection.

Authentication and Authorization:
- Verify that authentication mechanisms are implemented correctly and user credentials are securely stored and transmitted.
- Ensure that authorization checks are in place to restrict access to sensitive functions and data.

Data Sanitization and Encoding:
- Check if user input and other data are sanitized and properly encoded when being displayed or stored to prevent data corruption and injection attacks.

Session Management:
- Evaluate the implementation of session management to ensure secure session handling, token management, and session expiration.

Error Handling and Logging:
- Review error handling mechanisms to avoid leaking sensitive information and ensure proper error logging for debugging and auditing purposes.

Cryptography:
- Check the usage of cryptographic functions and algorithms to ensure they are implemented correctly and securely.

Secure Communication:
- Verify that sensitive information is transmitted over secure channels (e.g., HTTPS) and that appropriate encryption is used for data in transit.

Cross-Site Scripting (XSS) Prevention:
- Inspect the code for proper sanitization and encoding of user-generated content to prevent XSS attacks.

Cross-Site Request Forgery (CSRF) Prevention:
- Check if measures are in place to protect against CSRF attacks, such as using anti-CSRF tokens and proper validation of requests.

Secure File Handling:
- Review file upload and download functionality to ensure secure file handling practices, including proper validation, sanitization, and secure storage.

Code Injection Prevention:

- Check for vulnerabilities related to code injection attacks, such as ensuring user input is not executed as code or commands directly.

Third-Party Libraries and Dependencies:

- Assess the usage of third-party libraries and dependencies, ensuring they are up to date and free from known security vulnerabilities.

Secure Configuration:

- Review configuration files and settings to ensure secure configurations are applied, such as disabling unnecessary services and using strong passwords.

Security Headers and Best Practices:

- Check if security headers, such as Content Security Policy (CSP) and HTTP Strict Transport Security (HSTS), are properly implemented.

Documentation and Security Guidelines:

- Evaluate the presence of secure coding guidelines and documentation to ensure that developers follow best practices and security standards.