www.certifiedsecure.com
info@certifiedsecure.com
Tel.: +31 (0)70 310 13 40
Loire 128-A
2491 AJ The Hague
The Netherlands

Certified Secure Web Application Secure Development Checklist

About

Certified Secure exists to encourage and fulfill the growing interest in IT security knowledge and skills. We stand for openness, transparency and the sharing of knowledge; making sure everybody can experience and enjoy IT security. Security is serious fun!

All Certified Secure certifications, products and training are developed by IT security professionals with international recognized expertise. Our involvement in the IT security community worldwide, ensures relevant and high-quality standards. Delivering a wide variety of online challenges, videos, tools and more, Certified Secure is the authoritative source for practical IT security know-how.

Scope

This checklist can be used as a standard when developing or auditing a web based application. For security testers a separate Web Application Security Test Checklist is available from https://www.certifiedsecure.com/checklists.

Usage

Developers should use this checklist when developing a web application. Software auditors should use this checklist when performing a white box source code audit of a web application. A risk analysis for the web application should be performed before starting with the checklist. Every control on the checklist should be completed or explicitly marked as being not applicable. Once a control is completed the checklist should be updated with the appropriate result icon and a document cross-reference.

The completed checklist should never be delivered standalone but should be incorporated in a report detailing the risk analysis and checklist results. When performing a white box source code audit the report should also include the scope and context of the performed audit.

License

Result Icon Legend

Icon	Explanation
	Test was performed and results are okay
	Test was performed and results require attention
	Test was not applicable

#	Certified Secure Web Application Secure Development Checklist	Result	Ref
1.0	**Generic**		
1.1	Write clean, maintainable and readable code		
1.2	Use assertions to double-check critical (security) assumptions		
1.3	Separate the presentation layer from the data/business logic layer		
2.0	**Deployment**		
2.1	Install security updates for all deployed software		
2.2	Never use unsupported or end-of-life software versions		
2.3	Disable the HTTP TRACK and TRACE methods		
2.4	Remove all extraneous functionality		
2.5	Complete the Server Configuration Checklist for all servers		
3.0	**Information Disclosure**		
3.1	Remove all extraneous files and directories		
3.2	Prevent extraneous directory listings		
3.3	Implement access controls for debug functionality		
3.4	Only display generalized log and error messages to users		
3.5	Prevent direct propagation of errors and exceptions to users		
3.6	Never store sensitive information in the robots.txt file		
3.7	Remove sensitive information from publicly accessible source code		
3.8	Never disclose internal addresses		

#	Certified Secure Web Application Secure Development Checklist	Result	Ref
4.0	**Privacy and Confidentiality**		
4.1	Never store sensitive information in URLs		
4.2	Never store unencrypted sensitive information at the client-side		
4.3	Disallow indexing of pages with sensitive information		
4.4	Never include content from untrusted (external) sources		
4.5	Implement anti-caching measures for pages with sensitive information		
4.6	Transmit all sensitive information via secure connections		
4.7	Deny access to sensitive information via insecure connections		
4.8	Configure SSL/TLS for all pages on sites processing sensitive information		
4.9	Never include non-SSL/TLS resources in SSL/TLS pages		
4.10	Emit the HSTS header for full SSL sites		
4.11	Never use untrusted or expired SSL certificates		
4.12	Configure SSL/TLS to use only strong keys, ciphers and protocols		
4.13	Use only widely accepted and proven cryptographic primitives		
4.14	Use existing, well-tested implementations of cryptographic primitives		
5.0	**State Management**		
5.1	Implement state management at the server-side		
5.2	Explicitly validate all state transitions		
6.0	**Authentication and Authorization**		
6.1	Implement a default-deny policy for all authorization mechanisms		

#	Certified Secure Web Application Secure Development Checklist	Result	Ref
6.2	Authorize and authenticate every privileged operation		
6.3	Implement authorization and authentication at the server-side		
6.4	Implement authorization and authentication at a centralized location		
6.5	Remove all test and default accounts from production systems		
6.6	Use a cryptographically strong PRNG for generating tokens		
6.7	Use a sufficiently large size for generated tokens (at least 128 bits)		
6.8	Never implement authentication or authorization based on obscurity		
6.9	Enforce the usage of strong passwords		
6.10	Sufficiently hash, salt and stretch stored passwords		
6.11	Implement rate limiting for all authentication functionality		
6.12	Require re-authentication when changing credentials		
6.13	Always implement logout functionality		
7.0	**User Input**		
7.1	Encode user input in the context where it is used		
7.2	Accept only specific data types (string, integer, etc.) for user input		
7.3	Never process unexpected user input		
7.4	Use parameterized SQL queries		
7.5	Encode user input before it is used in SQL statements		
7.6	Encode user input before it is used in filenames or directories		
7.7	Encode user input before it is used in a HTML context		

#	Certified Secure Web Application Secure Development Checklist	Result	Ref
7.8	Encode user input before it is used in a JavaScript context		
7.9	Encode user input before it is used in system commands		
7.10	Encode user input before it is used in XML documents		
7.11	Encode user input before it is used in XPath queries		
7.12	Encode user input before it is used in XSL(T) style sheets		
7.13	Encode user input before it is used in SSI statements		
7.14	Encode user input before it is used in HTTP headers		
7.15	Encode user input before it is used in HTTP parameters		
7.16	Encode user input before it is used in LDAP queries		
7.17	Encode user input before it is used in regular expressions		
7.18	Encode user input before it is used in eval functions		
7.19	Encode user input before it is used in expression languages		
7.20	Encode user input before it is used in other protocols		
8.0	**Sessions**		
8.1	Deny all data/state mutation requests without a valid CSRF token		
8.2	Deny all privileged operation requests without a valid CSRF token		
8.3	Use a cryptographically strong PRNG for generating CSRF tokens		
8.4	Use a sufficiently large size for CSRF tokens (at least 128 bits)		
8.5	Revoke the session on logout		
8.6	Regenerate the session identifier on login		

#	Certified Secure Web Application Secure Development Checklist	Result	Ref
8.7	Regenerate the session identifier when changing credentials		
8.8	Revoke other sessions when changing credentials		
8.9	Set the Secure flag for session cookies		
8.10	Set the HttpOnly flag for session cookies		
8.11	Set a restrictive domain for session cookies		
8.12	Set a restrictive path for session cookies		
8.13	Use a cryptographically strong PRNG for generating session identifiers		
8.14	Use a sufficiently large size for session identifiers (at least 128 bits)		
8.15	Reject unknown session identifiers received from the client-side		
8.16	Only transmit session identifiers via secure connections		
8.17	Bind sessions to a specific IP address		
8.18	Enforce periodic session expiry and revocation		
9.0	**File Uploads**		
9.1	Store uploaded files outside the document root		
9.2	Never execute or evaluate uploaded files		
9.3	Enforce a conservative file size limit for uploaded files		
9.4	Accept only specific file types (.png, .txt, etc.) for uploaded files		
10.0	**Content**		
10.1	Explicitly define a specific content type for all served resources		
10.2	Explicitly define a character set for all served resources		

#	Certified Secure Web Application Secure Development Checklist	Result	Ref
10.3	Implement anti content sniffing measures for all served resources		
11.0	**XML Processing**		
11.1	Disable XML external entity expansion		
11.2	Disable XML external DTD parsing		
11.3	Disable all extraneous or dangerous XML extensions		
11.4	Use an XML parser that does not recursively expand entities		
12.0	**Miscellaneous**		
12.1	Implement anti-clickjacking measures		
12.2	Never redirect to user provided external locations		
12.3	Restrict cross-domain access using a whitelist based approach		
12.4	Implement rate limiting for all e-mail based functionality		
12.5	Implement rate limiting for all resource intensive functionality		
12.6	Prevent unintended denial of service when implementing rate limiting		
12.7	Check for and mitigate application- or setup-specific problems		