# Application Security Testing

**Assessment, Review & Guidelines**

DREW
Blandford-Williams

GRC

cybersecurity consulting services

| | |
|---|---|
| **Document Audience** | **Application Developers** |
| **Document Description** | The Vendor-Technology Application Security Checklist is a combination of many OWASP and SANS documents included below and aims to help developers evaluate their coding from a security perspective. This document is focused on secure coding requirements rather than specific vulnerabilities. It is more focused on web application programming although one can also use many of these practices for traditional desktop, mobile, or legacy software.<br>• OWASP Top 10 Application Security Vulnerabilities (2013)<br>• CWE/SANS Top 25 Software Errors (2011)<br>• OWASP & CWE/SANS Crosswalk Mapping<br>• OWASP Secure Coding Practice Guide V2.0<br>• OWASP Code Review Guide V2.0<br>• OWASP Test Guide V4.0<br>• OWASP Application Security Verification Standard 2014 |

## Application Name:

### Related SRAQ:

*(Related SRAQ Name/URL)*

### Application Language/Platform Description:

*(Java, .NET, Ruby, PHP, Rails, Spring, Web-based, Client-Server, Windows, LAMP, etc)*

### Attack Surface Description:

*(Enumerate all of the entry points in the code an attacker could attempt to exploit. Examples: standard web form URLs, AJAX URLs, web services, data feeds, service bus messages, etc.  Consider the entire attack surface when reviewing requirements below.)*

### Review Performed By:

*(Name, Date)*

| | |
|---|---|
| **1** | **Input Validation**<br>Failure to properly server-side validate input data from untrusted sources is the most common application security weakness and it can lead to major vulnerabilities in applications such as cross-site scripting (XSS), SQL injection, buffer overflow, etc. Bad input can also lead to Denial of Service (DoS) attacks on the application. As such it is important to always validate input data based on data type and range. Rather than using blacklist techniques to filter out bad input, it is recommended to use whitelist techniques to accept only allowed characters or values as valid input. JavaScript/client-side validation alone is not adequate. |
| **Input Validation related OWASP Top 10 and CWE/SANS Top 25 Elements** | • OWASP Top 10: A1 - Injection<br>• OWASP Top 10: A3 - Cross-Site Scripting<br>• OWASP Top 10: A10 - Unvalidated Redirects and Forwards<br>• CWE-20: Improper Input Validation<br>• CWE-89: SQL Injection<br>• CWE-91: XML Injection<br>• CWE-90: LDAP Injection<br>• CWE-98: Remote File Inclusion<br>• CWE-78: OS Command Injection<br>• CWE-120: Buffer Overflow<br>• CWE-22: Path Traversal<br>• CWE-79: Cross-Site Scripting<br>• CWE-601: URL Redirection to Untrusted Site<br>• CWE-807: Reliance on Untrusted Inputs<br>• CWE-131: Incorrect Calculation of Buffer Size<br>• CWE-134: Uncontrolled Format String<br>• CWE-190: Integer Overflow or Wraparound<br>• CWE-676: Use of Potentially Dangerous Function |
| **Coding Examples & Reference Materials** | • OWASP - Input Validation Cheat Sheet<br>• OWASP – 2014 Top Ten Proactive Controls for Application Security<br>• OWASP – Testing for Input Validation<br>• CWE – Improper Input Validation<br>• CWE – Establish and Maintain Control over all of your Inputs |
| **How are you addressing Input Validation for your application?** | **Status** |
| **Comments:**<br>*Comments Here* | *Select One* |

| | |
|---|---|
| **2** | **Output Escaping/Encoding**<br>Output escaping/encoding is how an application handles output. Output can often contain input data supplied from users, databases, external systems, etc. Secure output handling is often associated with preventing cross-site scripting and its purpose (as it relates to security) is to convert untrusted input into a safe form where the input is displayed as data to the user without executing as code in the destination (i.e. browser, database, OS). Escape/encode all output data unless they are known to be safe for the intended destination. Consider also implementing Content Security Policy (CSP) if possible. |
| **Output Escaping/Encoding related OWASP Top 10 and CWE/SANS Top 25 Elements** | • OWASP Top 10: A3 - Cross-Site Scripting<br>• CWE-79: Cross-Site Scripting<br>• CWE-601: URL Redirection to Untrusted Site |
| **Coding Examples & Reference Materials** | • OWASP – 2014 Top Ten Proactive Controls for Application Security<br>• CWE – Improper Encoding or Escaping of Output<br>• CWE - Establish and Maintain Control over all your Outputs<br>• Output Encoding: XSS Prevention Cheat Sheet<br>• Output Encoding: SQL Injection Prevention Cheat Sheet<br>• Output Encoding: Preventing OS Injection<br>• Content Security Policy (CSP) |
| **How are you addressing Output Escaping/Encoding for your application?** | **Status** |
| **Comments:**<br>*Comments Here* | *Select One* |

| | |
|---|---|
| **3** | **Authentication & Password Management**<br>Authentication is the process of verifying that an individual or entity is who they claim to be. Proper use of an external centralized authentication system should significantly reduce the likelihood of a problem in this area. Create a password policy to document and address key concerns when it comes to authentication and password management including proper password strength controls, password lifecycle, password reset process, password storage, protecting credentials in transit, browser caching, number of login attempts, etc. For unauthenticated/anonymous page submits, consider using CAPTCHA technology to prevent spam and automated attacks. Enforce multi-factor authentication in high risk areas where possible.<br>In the case of application authenticating to external systems (like databases, file servers, web services), the credentials should be encrypted at rest with proper access controls and never stored in source code. |
| **Authentication & Password Management related OWASP Top 10 and CWE/SANS Top 25 Elements** | • OWASP Top 10: A2 - Broken Authentication and Session Management<br>• OWASP Top 10: A8 - Cross-Site Request Forgery (CSRF)<br>• CWE-287: Improper Authentication<br>• CWE-306: Missing Authentication for Critical Function<br>• CWE-307: Improper Restriction of Excessive Authentication Attempts<br>• CWE-352: Cross-Site Request Forgery (CSRF)<br>• CWE-798: Use of Hard-Coded Credentials |
| **Coding Examples & Reference Materials** | • OWASP – Authentication Cheat Sheet<br>○ Authentication: Forgot Password Cheat Sheet<br>○ Authentication: Password Storage Cheat Sheet<br>• OWASP – 2014 Top Ten Proactive Controls for Application Security<br>• CWE – Industry Accepted Security Features<br>• Secure Coding Cheat Sheet - Authentication & Password Management |
| **How are you addressing Authentication & Password Management for your application?** | **Status** |
| **Comments:**<br>*Comments Here* | *Select One* |

| | |
|---|---|
| **4** | **Session Management**<br>Session management ensures that authenticated users have a robust and cryptographically secure association with their session. It is recommended to use the server or framework's session management controls whenever possible. Also the following areas should be considered: session invalidation during authentication, re-authentication, logout, and switching from HTTPS to HTTP. HTTP header tags like timeout, domain, path, http only, and secure should also be considered with regards to session management. If using single-sign-on, make sure the application logout function calls the single-sign-on logout function.  Force user re-verification, not relying only on current session state, for high-risk user transactions to prevent CSRF. |
| **Session Management related OWASP Top 10 and CWE/SANS Top 25 Elements** | • OWASP Top 10: A2 - Broken Authentication and Session Management<br>• OWASP Top 10: A8 - Cross-Site Request Forgery (CSRF)<br>• CWE-384: Session Fixation<br>• CWE-613: Insufficient Session Expiration<br>• CWE-287: Improper Authentication<br>• CWE-306: Missing Authentication for Critical Function<br>• CWE-307: Improper Restriction of Excessive Authentication Attempts<br>• CWE-352: Cross-Site Request Forgery (CSRF)<br>• CWE-798: Use of Hard-Coded Credentials |
| **Coding Examples & Reference Materials** | • OWASP – Session Management Cheat Sheet<br>• OWASP – Session Management 2009 Version<br>• OWASP – 2014 Top Ten Proactive Controls for Application Security<br>• CWE – Industry Accepted Security Features<br>• Secure Coding Cheat Sheet - Session Management |
| **How are you addressing Session Management for your application?** | **Status** |
| **Comments:**<br>*Comments Here* | *Select One* |

| | |
|---|---|
| **5** | **Authorization & Access Control**<br>Once an identity (subject) is authenticated, authorization is the decision process where requests to (create, read, update, delete, etc) a particular resource (object) should be granted or denied. Access control is the method used for authorization enforcement with the most popular being role-based access control (RBAC). It is preferred to use an external centralized authorization system where role membership is centrally managed and audited, then map those roles to specific permissions within the application.<br>Implement least privilege policy between all subjects and objects. Ensure that the access control list covers all possible scenarios. Enforce timely authorization checks on every request (from both server and client side) to prevent "time of check"/"time of use" (TOC/TOU) attacks. |
| **Authorization & Access Control related OWASP Top 10 and CWE/SANS Top 25 Elements** | • OWASP Top 10: A4 - Insecure Direct Object References<br>• OWASP Top 10: A7 - Missing Function Level Access Control<br>• CWE-22: Path Traversal<br>• CWE-250: Execution with Unnecessary Privileges<br>• CWE-434: Unrestricted Upload of File with Dangerous Type<br>• CWE-829: Inclusion of Functionality from Untrusted Control Sphere<br>• CWE-862: Missing Authorization<br>• CWE-863: Incorrect Authorization<br>• CWE-732: Incorrect Permission Assignment for Critical Resource |
| **Coding Examples & Reference Materials** | • OWASP - Access Control Cheat Sheet<br>• OWASP – 2014 Top Ten Proactive Controls for Application Security<br>• CWE – Industry Accepted Security Features<br>• Secure Coding Cheat Sheet - Access Control |
| **How are you addressing Authorization & Access Control for your application?** | **Status** |
| **Comments:**<br>*Comments Here* | *Select One* |

| 6 | **Cryptographic Practices**<br>Proper encryption should be used when handling sensitive data at any tier of the application. Choose carefully whether “two-way” shared key symmetric encryption, “two-way” public/private key asymmetric encryption, or “one-way” salted hash encryption is best for each case. Ensure cryptographic modules used by the application are compliant with FIPS 140-2 or an equivalent standard (see Module Validation Lists) both from vendor and algorithm perspectives. Only use approved cryptographic modules for random number generators. |
|---|---|
| **Cryptographic Practices related OWASP Top 10 and CWE/SANS Top 25 Elements** | • CWE-311: Missing Encryption of Sensitive Data<br>• CWE-327: Use of a Broken or Risky Cryptographic Algorithm<br>• CWE-759: Use of a One-Way Hash without a Salt |
| **Coding Examples & Reference Materials** | • OWASP – Cryptographic Storage Cheat Sheet<br>• OWASP – User Privacy Protection Cheat Sheet<br>• OWASP – 2014 Top Ten Proactive Controls for Application Security<br>• OWASP – Guide to Cryptography<br>• CWE – Industry Accepted Security Features |
| **How are you addressing Cryptographic Practices for your application?** | **Status** |
| **Comments:**<br>*Comments Here* | *Select One* |

| 7 | **Error Handling, Auditing & Logging**<br>The application should handle its own application errors and not rely on the server. Do not display sensitive, debug or stack trace information in the production environment. Ensure audit logging controls are in place to log both successful/failure security events, especially authentication/authorization attempts and access to sensitive data with useful audit information based on the "Who/What/When/Where" principal. Sensitive data should never be logged, instead use other unique and traceable identifiers. |
|---|---|
| **Error Handling, Auditing & Logging related OWASP Top 10 and CWE/SANS Top 25 Elements** | • CWE-754: Improper Check for Unusual or Exceptional Conditions<br>• CWE-209: Information Exposure Through an Error Message<br>• CWE-306: Missing Authentication for Critical Function<br>• CWE-862: Missing Authorization |
| **Coding Examples & Reference Materials** | • OWASP – Error Handling, Auditing & Logging<br>• OWASP – Logging Cheat Sheet<br>• OWASP – 2014 Top Ten Proactive Controls for Application Security<br>• CWE – Industry Accepted Security Features |

| **How are you addressing Error Handling, Auditing & Logging for your application?** | **Status** |
|---|---|
| **Comments:**<br>*Comments Here* | *Select One* |

| | |
|---|---|
| **8** | **Data Protection**<br>Limit access to data based on the least privilege principal. Encrypt sensitive data and information like stored passwords, connection strings and properly protect decryption keys. Make sure all cached or temporary copies of sensitive data are protected from unauthorized access and get purged as soon as they are no longer required. Do not allow sensitive production data in non-production environments. Do not include sensitive information in HTTP GET URL. Consider using the following HTTP headers: Cache-Control: no-cache, no-store; Expires: 0 and Cache-Control: max-age=0. |
| **Data Protection related OWASP Top 10 and CWE/SANS Top 25 Elements** | • OWASP Top 10: A6 - Sensitive Data Exposure<br>• CWE-311: Missing Encryption of Sensitive Data<br>• CWE-327: Use of a Broken or Risky Cryptographic Algorithm<br>• CWE-759: Use of a One-Way Hash without a Salt |
| **Coding Examples & Reference Materials** | • OWASP – Cryptographic Storage Cheat Sheet<br>• OWASP – User Privacy Protection Cheat Sheet<br>• OWASP – Password Storage Cheat Sheet<br>• OWASP – 2014 Top Ten Proactive Controls for Application Security<br>• OWASP – Testing Browser Cache Weakness<br>• CWE – Industry Accepted Security Features |
| **How are you addressing Data Protection for your application?** | **Status** |
| **Comments:**<br>*Comments Here* | *Select One* |

| | |
|---|---|
| **9** | **Communication Security**<br>When transmitting sensitive information, at any tier of the application or network architecture, encryption-in-transit should be used. SSL/TLS is by far the most common and widely supported model. Use a trusted certificate authority to generate public and private keys whenever possible. In the case of using in-house CA make sure proper security controls are in place to protect the private keys from unauthorized access. Make sure that the server only supports approved strong cipher modules. |
| **Communication Security related OWASP Top 10 and CWE/SANS Top 25 Elements** | • OWASP Top 10: A6 - Sensitive Data Exposure<br>• CWE-311: Missing Encryption of Sensitive Data<br>• CWE-327: Use of a Broken or Risky Cryptographic Algorithm<br>• CWE-759: Use of a One-Way Hash without a Salt |
| **Coding Examples & Reference Materials** | • OWASP – Transport Layer Protection Cheat Sheet<br>• Secure Coding Cheat Sheet – Secure Transmission<br>• OWASP – Testing for SSL-TLS<br>• OWASP – 2014 Top Ten Proactive Controls for Application Security<br>• OWASP – Guide to Cryptography<br>• CWE – Industry Accepted Security Features |
| **How are you addressing Communication Security for your application?** | **Status** |
| **Comments:**<br>*Comments Here* | *Select One* |

| | |
|---|---|
| **10** | **System Configuration/Hardening**<br>Make sure that every piece of software from the OS, system components, software libraries, software framework, web servers, etc. are running the latest version and they are patched with latest security patches. Lock down the server and remove any unnecessary files and functions. Isolate development environments from production environments. Use version control software so that all code changes deployed to production are reviewed and have an audit trail. |
| **System Configuration related OWASP Top 10 and CWE/SANS Top 25 Elements** | • OWASP Top 10: A5 - Security Misconfiguration<br>• OWASP Top 10: A9 - Using Components with Known Vulnerabilities<br>• CWE-250: Execution with Unnecessary Privileges<br>• CWE-732: Incorrect Permission Assignment for Critical Resource<br>• CWE-494: Download of Code Without Integrity Check<br>• CWE-829: Inclusion of Functionality from Untrusted Control Sphere |
| **Coding Examples & Reference Materials** | • OWASP – Testing for Configuration Management<br>• OWASP – Configuration Guide |

| How are you addressing System Configuration for your application? | Status |
|---|---|
| **Comments:**<br>*Comments Here* | *Select One* |

| **11** | **Database Security**<br>Use parameterized queries even if using a popular data persistence layer like Hibernate or .Net Entity Framework. Don't try to build dynamic SQL queries. The application should use the lowest possible level of privilege when accessing the database. Lock down the database by turning off any unnecessary features. Connection strings and database passwords should not be hard coded within the application. Keep them in secure, separate and encrypted configuration files. |
|---|---|
| **Database Security related OWASP Top 10 and CWE/SANS Top 25 Elements** | • OWASP Top 10: A1 - Injection<br>• CWE-22: Path Traversal<br>• CWE-89: SQL Injection<br>• CWE-732: Incorrect Permission Assignment for Critical Resource<br>• CWE-759: Use of a One-Way Hash without a Salt<br>• CWE-863: Incorrect Authorization |
| **Coding Examples & Reference Materials** | • OWASP – Configuration Guide<br>• OWASP – Secure Coding Practices<br>• OWASP – 2014 Top Ten Proactive Controls for Application Security |
| **How are you addressing Database Security for your application?** | **Status** |
| **Comments:**<br>*Comments Here* | *Select One* |

| | |
|---|---|
| **12** | **File Management**<br>Ensure authentication is required before file uploads. Limit file types & prevent any file types that may be interpreted by the web server as well as validate the file types by checking the file header. Do not save the uploaded file in the same web context as the application. Do not pass directory or file paths to the user, use index values mapped to pre-defined paths. Never send absolute file path to client. Scan uploaded files for malware where possible. |
| **File Management related OWASP Top 10 and CWE/SANS Top 25 Elements** | • OWASP Top 10: A4 - Insecure Direct Object References<br>• CWE-287: Improper Authentication<br>• CWE-306: Missing Authentication for Critical Function<br>• CWE-307: Improper Restriction of Excessive Authentication Attempts<br>• CWE-434: Unrestricted Upload of File with Dangerous Type |
| **Coding Examples & Reference Materials** | • OWASP – File System Management<br>• OWASP – 2014 Top Ten Proactive Controls for Application Security<br>• CWE – Industry Accepted Security Features |
| **How are you addressing File Management for your application?** | **Status** |
| **Comments:**<br>*Comments Here* | *Select One* |