Application Configuration Standard

*Revision: r1.0*

*Effective Date: ddmmyyyy*

*Classification: Internal*

**Internal INFORMATION**

This is a proprietary document and is the property of XXXX.; it contains information that is proprietary, or otherwise restricted from disclosure. If you are not an authorised recipient, please return this document to the above-named owner(s). Dissemination, distribution, copying or use of this document in whole or in part by anyone other than the intended recipient is strictly prohibited without prior written permission of XXXX.

Table of Contents

1. Introduction 4

1.1 Document Definition 4

1.2 Objective 4

1.3 Scope 4

1.3.1 Applicability to employees 4

1.3.1 Applicability to External Parties 4

1.3.2 Applicability to Assets 4

1.4 Industry Configuration Standards 4

1.5 Related Documents / References 4

2. Standard Statements 5

2.1 Application Security Verification (AVS) Levels 5

2.2 Architecture, design and threat modelling 5

2.3 Authentication Verification Requirements 6

2.4 Session Management Verification Requirements 8

2.5 Access Control Verification Requirements 8

2.6 Malicious input handling verification requirements 9

2.7 Cryptography at rest verification requirements 11

2.8 Error handling and logging verification requirements 12

2.9 Data protection verification requirements 12

2.10 Communications security verification requirements 13

2.11 HTTP security configuration verification requirements 15

2.12 Malicious controls verification requirements 15

2.13 Business logic verification requirements 15

2.14 Files and resources verification requirements 16

2.15 Mobile verification requirements 17

2.16 Web services verification requirements 18

2.17 Configuration 18

3. Standard Compliance & Enforcement 20

3.1 Compliance Measures 20

3.2 Enforcement 20

4. Exception Process / Glossary 21

4.1 Exception Process 21

4.2 Glossary / Acronyms 21

5. Document Management 22

5.1 Document Revision Log 22

5.2 Document Ownership 22

5.3 Document Coordinator 22

5.4 Document Approvers 22

5.5 Distribution 22

# Introduction

## Document Definition

This document is a Standard.

For a full description of document types, see XXXX-POL-ALL-001 - Information Security Policy Framework.

## Objective

The objective of this standard is to ensure the XXXXs business and technical requirements for all applications (purchased, leased, open source, in-house developed or cloud-based applications) are translated into a baseline that will allow administrators to easily configure the application to adequately protect the XXXX’s information assets against external and internal threats.

## Scope

### Applicability to employees

XXXX refers to XXXX as well as its majority-owned subsidiaries and joint ventures (if applicable). This Standard applies to all employees, officers, members of Board of Directors, and all consultants, and contractors.

### Applicability to External Parties

Relevant Standard statements will apply to any external party and be included in contractual obligations on a case-by-case basis.

### Applicability to Assets

This Standard applies to all information assets globally owned by XXXX, or where XXXX has custodial responsibilities.

## Industry Configuration Standards

* Centre for Internet Security (CIS): http://www.cisecurity.org/
* Open Web Application Security Project (OWASP) Application Security Verification Standard 3.0

## Related Documents / References

* *XXXX-POL-ALL-001 - Information Security Policy Framework*

# Standard Statements

## Application Security Verification (AVS) Levels

Each ASV level contains a list of security requirements. Each of these requirements should be mapped to

security-specific features and capabilities that must be built into each application.

Level 1 is meant for all applications.

Level 2 is for applications that contain sensitive data, which requires protection.

Level 3 is for the most critical applications - applications that perform high value transactions, or any application that requires the highest level of trust.

## Architecture, design and threat modelling

Requirements

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | | **Description** | | **1** | | **2** | | **3** | |
| **1.1** | | Verify that all application components are identified and are known to be needed. | | ✓ | | ✓ | | ✓ | |
| **1.2** | | Verify that all components, such as libraries, modules, and external systems, that are not part of the application but that the application relies on to operate are identified. | |  | | ✓ | | ✓ | |
| **1.3** | | Verify that a high-level architecture for the application has been defined. | |  | | ✓ | | ✓ | |
| **1.4** | | Verify that all application components are defined in terms of the business functions and/or security functions they provide. | |  | |  | | ✓ | |
| **1.5** | | Verify that all components that are not part of the application but that the application relies on to operate are defined in terms of the functions, and/or security functions, they provide. | |  | |  | | ✓ | |
| **1.6** | | Verify that a threat model for the target application has been produced and covers off risks associated with Spoofing, Tampering, Repudiation, Information Disclosure, and Elevation of privilege (STRIDE). | |  | |  | | ✓ | |
| **1.7** | | Verify all security controls (including libraries that call external security services) have a centralized implementation. | |  | |  | | ✓ | |
| **1.8** | | Verify that components are segregated from each other via a defined security control, such as network segmentation, firewall rules, or cloud based security groups. | |  | | ✓ | | ✓ | |
| **1.9** | | Verify the application has a clear separation between the data layer, controller layer and the display layer, such that security decisions can be enforced on trusted systems. | |  | | ✓ | | ✓ | |
| Verify that there is no sensitive business logic, secret keys or other proprietary information in client side code. | |  | | ✓ | | ✓ | |

## Authentication Verification Requirements

Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **1** | **2** | **3** |
| **2.1** | Verify all pages and resources by default require authentication except those specifically intended to be public (Principle of complete mediation). | ✓ | ✓ | ✓ |
| **2.2** | Verify that all password fields do not echo the user’s password when it is entered. | ✓ | ✓ | ✓ |
| **2.4** | Verify all authentication controls are enforced on the server side. | ✓ | ✓ | ✓ |
| **2.6** | Verify all authentication controls fail securely to ensure attackers cannot log in. | ✓ | ✓ | ✓ |
| **2.7** | Verify password entry fields allow, or encourage, the use of passphrases, and do not prevent long passphrases/highly complex passwords being entered. | ✓ | ✓ | ✓ |
| **2.8** | Verify all account identity authentication functions (such as update profile, forgot password, disabled / lost token, help desk or IVR) that might regain access to the account are at least as resistant to attack as the primary authentication mechanism. | ✓ | ✓ | ✓ |
| **2.9** | Verify that the changing password functionality includes the old password, the new password, and a password confirmation. | ✓ | ✓ | ✓ |
| **2.12** | Verify that all suspicious authentication decisions are logged. This should include requests with relevant metadata needed for security investigations. |  | ✓ | ✓ |
| **2.13** | Verify that account passwords make use of a sufficient strength encryption routine and that it withstands brute force attack against the encryption routine. |  | ✓ | ✓ |
| **2.16** | Verify that credentials are transported using a suitable encrypted link and that all pages/functions that require a user to enter credentials are done so using an encrypted link. | ✓ | ✓ | ✓ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **2.17** | Verify that the forgotten password function and other recovery paths do not reveal the current password and that the new password is not sent in clear text to the user. | ✓ | ✓ | ✓ |
| **2.18** | Verify that information enumeration is not possible via login, password reset, or forgot account functionality. | ✓ | ✓ | ✓ |
| **2.19** | Verify there are no default passwords in use for the application framework or any components used by the application (such as “admin/password”). | ✓ | ✓ | ✓ |
| **2.20** | Verify that request throttling is in place to prevent automated attacks against common authentication attacks such as brute force attacks or denial of service attacks. | ✓ | ✓ | ✓ |
| **2.21** | Verify that all authentication credentials for accessing services external to the application are encrypted and stored in a protected location. |  | ✓ | ✓ |
| **2.22** | Verify that forgotten password and other recovery paths use a soft token, mobile push, or an offline recovery mechanism. | ✓ | ✓ | ✓ |
| **2.23** | Verify that account lockout is divided into soft and hard lock status, and these are not mutually exclusive. If an account is temporarily soft locked out due to a brute force attack, this should not reset the hard lock status. |  | ✓ | ✓ |
| **2.24** | Verify that if knowledge based questions (also known as "secret questions") are required, the questions should be strong enough to protect the application. | ✓ | ✓ | ✓ |
| **2.25** | Verify that the system can be configured to disallow the use of a configurable number of previous passwords. |  | ✓ | ✓ |
| **2.26** | Verify re-authentication, step up or adaptive authentication, two factor authentication, or transaction signing is required before any application-specific sensitive operations are permitted as per the risk profile of the application. |  | ✓ | ✓ |
| **2.27** | Verify that measures are in place to block the use of commonly chosen passwords and weak passphrases. | ✓ | ✓ | ✓ |
| **2.28** | Verify that all authentication challenges, whether successful or failed, should respond in the same average response time. |  |  | ✓ |
| **2.29** | Verify that secrets, API keys, and passwords are not included in the source code, or online source code repositories. |  |  | ✓ |
| **2.30** | Verify that if an application allows users to authenticate, they use a proven secure authentication mechanism. | ✓ | ✓ | ✓ |
| **2.31** | Verify that if an application allows users to authenticate, they can authenticate using two-factor authentication or other strong authentication, or any similar scheme that provides protection against username + password disclosure. |  | ✓ | ✓ |
| **2.32** | Verify that administrative interfaces are not accessible to untrusted parties | ✓ | ✓ | ✓ |

## Session Management Verification Requirements

Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **1** | **2** | **3** |
| **3.1** | Verify that there is no custom session manager, or that the custom session manager is resistant against all common session management attacks. | ✓ | ✓ | ✓ |
| **3.2** | Verify that sessions are invalidated when the user logs out. | ✓ | ✓ | ✓ |
| **3.3** | Verify that sessions timeout after a specified period of inactivity. | ✓ | ✓ | ✓ |
| **3.4** | Verify that sessions timeout after an administratively-configurable maximum time period regardless of activity (an absolute timeout). |  |  | ✓ |
| **3.5** | Verify that all pages that require authentication have easy and visible access to logout functionality. | ✓ | ✓ | ✓ |
| **3.6** | Verify that the session id is never disclosed in URLs, error messages, or logs. This includes verifying that the application does not support URL rewriting of session cookies. | ✓ | ✓ | ✓ |
| **3.7** | Verify that all successful authentication and re-authentication generates a new session and session id. | ✓ | ✓ | ✓ |
| **3.10** | Verify that only session ids generated by the application framework are recognized as active by the application. |  | ✓ | ✓ |
| **3.11** | Verify that session ids are sufficiently long, random and unique across the correct active session base. | ✓ | ✓ | ✓ |
| **3.12** | Verify that session ids stored in cookies have their path set to an appropriately restrictive value for the application, and authentication session tokens additionally set the “HttpOnly” and “secure” attributes | ✓ | ✓ | ✓ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **3.16** | Verify that the application limits the number of active concurrent sessions. | ✓ | ✓ | ✓ |
| **3.17** | Verify that an active session list is displayed in the account profile or similar of each user. The user should be able to terminate any active session. | ✓ | ✓ | ✓ |
| **3.18** | Verify the user is prompted with the option to terminate all other active sessions after a successful change password process. | ✓ | ✓ | ✓ |

## Access Control Verification Requirements

Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **1** | **2** | **3** |
| **4.1** | Verify that the principle of least privilege exists - users should only be able to access functions, data files, URLs, controllers, services, and other resources, for which they possess specific authorization. This implies protection against spoofing and elevation of privilege. | ✓ | ✓ | ✓ |
| **4.4** | Verify that access to sensitive records is protected, such that only authorized objects or data is accessible to each user (for example, protect against users tampering with a parameter to see or alter another user's account). | ✓ | ✓ | ✓ |
| **4.5** | Verify that directory browsing is disabled unless deliberately desired. Additionally, applications should not allow discovery or disclosure of file or directory metadata, such as Thumbs.db, .DS\_Store, .git or .svn folders. | ✓ | ✓ | ✓ |
| **4.8** | Verify that access controls fail securely. | ✓ | ✓ | ✓ |
| **4.9** | Verify that the same access control rules implied by the presentation layer are enforced on the server side. | ✓ | ✓ | ✓ |
| **4.10** | Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized. |  | ✓ | ✓ |
| **4.11** | Verify that there is a centralized mechanism (including libraries that call external authorization services) for protecting access to each type of protected resource. |  |  | ✓ |
| **4.12** | Verify that all access control decisions can be logged and all failed decisions are logged. |  | ✓ | ✓ |
| **4.13** | Verify that the application or framework uses strong random anti-CSRF tokens or has another transaction protection mechanism. | ✓ | ✓ | ✓ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4.14** | Verify the system can protect against aggregate or continuous access of secured functions, resources, or data. For example, consider the use of a resource governor to limit the number of edits per hour or to prevent the entire database from being scraped by an individual user. |  | ✓ | ✓ |
| **4.15** | Verify the application has additional authorization (such as step up or adaptive authentication) for lower value systems, and / or segregation of duties for high value applications to enforce anti-fraud controls as per the risk of application and past fraud. |  | ✓ | ✓ |
| **4.16** | Verify that the application correctly enforces context-sensitive authorisation so as to not allow unauthorised manipulation by means of parameter tampering. | ✓ | ✓ | ✓ |

## Malicious input handling verification requirements

Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **1** | **2** | **3** |
| **5.1** | Verify that the runtime environment is not susceptible to buffer overflows, or that security controls prevent buffer overflows. | ✓ | ✓ | ✓ |
| **5.3** | Verify that server side input validation failures result in request rejection and are logged. | ✓ | ✓ | ✓ |
| **5.5** | Verify that input validation routines are enforced on the server side. | ✓ | ✓ | ✓ |
| **5.6** | Verify that a single input validation control is used by the application for each type of data that is accepted. |  |  | ✓ |
| **5.10** | Verify that all SQL queries, HQL, OSQL, NOSQL and stored procedures, calling of stored procedures are protected by the use of prepared statements or query parameterization, and thus not susceptible to SQL injection | ✓ | ✓ | ✓ |
| **5.11** | Verify that the application is not susceptible to LDAP Injection, or that security controls prevent LDAP Injection. | ✓ | ✓ | ✓ |
| **5.12** | Verify that the application is not susceptible to OS Command Injection, or that security controls prevent OS Command Injection. | ✓ | ✓ | ✓ |
| **5.13** | Verify that the application is not susceptible to Remote File Inclusion (RFI) or Local File Inclusion (LFI) when content is used that is a path to a file. | ✓ | ✓ | ✓ |
| **5.14** | Verify that the application is not susceptible to common XML attacks, such as XPath query tampering, XML External Entity attacks, and XML injection attacks. | ✓ | ✓ | ✓ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **5.15** | Ensure that all string variables placed into HTML or other web client code is either properly contextually encoded manually, or utilize templates that automatically encode contextually to ensure the application is not susceptible to reflected, stored and DOM Cross-Site Scripting (XSS) attacks. | ✓ | ✓ | ✓ |
| **5.16** | If the application framework allows automatic mass parameter assignment (also called automatic variable binding) from the inbound request to a model, verify that security sensitive fields such as “*accountBalance*”, “*role*” or “*password*” are protected from malicious automatic binding. |  | ✓ | ✓ |
| **5.17** | Verify that the application has defenses against HTTP parameter pollution attacks, particularly if the application framework makes no distinction about the source of request parameters (GET, POST, cookies, headers, environment, etc.) |  | ✓ | ✓ |
| **5.18** | Verify that client side validation is used as a second line of defense, in addition to server side validation. |  | ✓ | ✓ |
| **5.19** | Verify that all input data is validated, not only HTML form fields but all sources of input such as REST calls, query parameters, HTTP headers, cookies, batch files, RSS feeds, etc; using positive validation (whitelisting), then lesser forms of validation such as greylisting (eliminating known bad strings), or rejecting bad inputs (blacklisting). |  | ✓ | ✓ |
| **5.20** | Verify that structured data is strongly typed and validated against a defined schema including allowed characters, length and pattern (e.g. credit card numbers or telephone, or validating that two related fields are reasonable, such as validating suburbs and zip or post codes match). |  | ✓ | ✓ |
| **5.21** | Verify that unstructured data is sanitized to enforce generic safety measures such as allowed characters and length, and characters potentially harmful in given context should be escaped (e.g. natural names with Unicode or apostrophes, such as ねこor O'Hara) |  | ✓ | ✓ |
| **5.22** | Make sure untrusted HTML from WYSIWYG editors or similar are properly sanitized with an HTML sanitizer and handle it appropriately according to the input validation task and encoding task. | ✓ | ✓ | ✓ |
| **5.23** | For auto-escaping template technology, if UI escaping is disabled, ensure that HTML sanitization is enabled instead. |  | ✓ | ✓ |
| **5.24** | Verify that data transferred from one DOM context to another, uses safe JavaScript methods, such as using .innerText and .val. |  | ✓ | ✓ |
| **5.25** | Verify when parsing JSON in browsers, that JSON.parse is used to parse JSON on the client. Do not use eval() to parse JSON on the client. |  | ✓ | ✓ |
| **5.26** | Verify that authenticated data is cleared from client storage, such as the browser DOM, after the session is terminated. |  | ✓ | ✓ |

## Cryptography at rest verification requirements

Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **1** | **2** | **3** |
| **7.2** | Verify that all cryptographic modules fail securely, and errors are handled in a way that does not enable oracle padding. | ✓ | ✓ | ✓ |
| **7.6** | Verify that all random numbers, random file names, random GUIDs, and random strings are generated using the cryptographic module’s approved random number generator when these random values are intended to be not guessable by an attacker. |  | ✓ | ✓ |
| **7.7** | Verify that cryptographic algorithms used by the application have been validated against FIPS 140-2 or an equivalent standard. | ✓ | ✓ | ✓ |
| **7.8** | Verify that cryptographic modules operate in their approved mode according to their published security policies. |  |  | ✓ |
| **7.9** | Verify that there is an explicit policy for how cryptographic keys are managed (e.g., generated, distributed, revoked, and expired). Verify that this key lifecycle is properly enforced. |  | ✓ | ✓ |
| **7.11** | Verify that all consumers of cryptographic services do not have direct access to key material. Isolate cryptographic processes, including master secrets and consider the use of a hardware key vault (HSM). |  |  | ✓ |
| **7.12** | *Personally Identifiable Information* should be stored encrypted at rest and ensure that communication goes via protected channels. |  | ✓ | ✓ |
| **7.13** | Verify that where possible, keys and secrets are zeroed when destroyed. |  | ✓ | ✓ |
| **7.14** | Verify that all keys and passwords are replaceable, and are generated or replaced at installation time. |  | ✓ | ✓ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **7.15** | Verify that random numbers are created with proper entropy even when the application is under heavy load, or that the application degrades gracefully in such circumstances. |  |  | ✓ |

## Error handling and logging verification requirements

Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **1** | **2** | **3** |
| **8.1** | Verify that the application does not output error messages or stack traces containing sensitive data that could assist an attacker, including session id, software/framework versions and personal information | ✓ | ✓ | ✓ |
| **8.2** | Verify that error handling logic in security controls denies access by default. |  | ✓ | ✓ |
| **8.3** | Verify security logging controls provide the ability to log success and particularly failure events that are identified as security-relevant. |  | ✓ | ✓ |
| **8.4** | Verify that each log event includes necessary information that would allow for a detailed investigation of the timeline when an event happens. |  | ✓ | ✓ |
| **8.5** | Verify that all events that include untrusted data will not execute as code in the intended log viewing software. |  |  | ✓ |
| **8.6** | Verify that security logs are protected from unauthorized access and modification. |  | ✓ | ✓ |
| **8.7** | Verify that the application does not log sensitive data as defined under local privacy laws or regulations, organizational sensitive data as defined by a risk assessment, or sensitive authentication data that could assist an attacker, including user’s session identifiers, passwords, hashes, or API tokens. |  | ✓ | ✓ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **8.8** | Verify that all non-printable symbols and field separators are properly encoded in log entries, to prevent log injection. |  |  | ✓ |
| **8.9** | Verify that log fields from trusted and untrusted sources are distinguishable in log entries. |  |  | ✓ |
| **8.10** | Verify that an audit log or similar allows for non-repudiation of key transactions. |  | ✓ | ✓ |
| **8.11** | Verify that security logs have some form of integrity checking or controls to prevent unauthorized modification. |  |  | ✓ |
| **8.12** | Verify that the logs are stored on a different partition than the application is running with proper log rotation. |  |  | ✓ |

## Data protection verification requirements

Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **1** | **2** | **3** |
| **9.1** | Verify that all forms containing sensitive information have disabled client side caching, including autocomplete features. | ✓ | ✓ | ✓ |
| **9.2** | Verify that the list of sensitive data processed by the application is identified, and that there is an explicit policy for how access to this data must be controlled, encrypted and enforced under relevant data protection directives. |  |  | ✓ |
| **9.3** | Verify that all sensitive data is sent to the server in the HTTP message body or headers (i.e., URL parameters are never used to send sensitive data). | ✓ | ✓ | ✓ |
| **9.4** | Verify that the application sets appropriate anti-caching headers as per the risk of the application, such as the following:  Expires: Tue, 03 Jul 2001 06:00:00 GMT Last-Modified: {now} GMT  Cache-Control: no-store, no-cache, must-revalidate, max-age=0 Cache-Control: post-check=0, pre-check=0  Pragma: no-cache | ✓ | ✓ | ✓ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **9.5** | Verify that on the server, all cached or temporary copies of sensitive data stored are protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data. |  | ✓ | ✓ |
| **9.6** | Verify that there is a method to remove each type of sensitive data from the application at the end of the required retention policy. |  |  | ✓ |
| **9.7** | Verify the application minimizes the number of parameters in a request, such as hidden fields, Ajax variables, cookies and header values. |  | ✓ | ✓ |
| **9.8** | Verify the application has the ability to detect and alert on abnormal numbers of requests for data harvesting for an example screen scraping. |  |  | ✓ |
| **9.9** | Verify that data stored in client side storage - such as HTML5 local storage, session storage, IndexedDB, regular cookies or Flash cookies - does not contain sensitive or PII). | ✓ | ✓ | ✓ |
| **9.10** | Verify accessing sensitive data is logged, if the data is collected under relevant data protection directives or where logging of accesses is required. |  | ✓ | ✓ |
| **9.11** | Verify that sensitive data is rapidly sanitized from memory as soon as it is no longer needed and handled in accordance to functions and techniques supported by the framework/library/operating system. |  | ✓ | ✓ |

## Communications security verification requirements

Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **1** | **2** | **3** |
| **10.1** | Verify that a path can be built from a trusted CA to each Transport Layer Security (TLS) server certificate, and that each server certificate is valid. | ✓ | ✓ | ✓ |
| **V10.3** | Verify that TLS is used for all connections (including both external and backend connections) that are authenticated or that involve sensitive data or functions, and does not fall back to insecure or unencrypted protocols. Ensure the strongest alternative is the preferred algorithm. | ✓ | ✓ | ✓ |
| **V10.4** | Verify that backend TLS connection failures are logged. |  |  | ✓ |
| **V10.5** | Verify that certificate paths are built and verified for all client certificates using configured trust anchors and revocation information. |  |  | ✓ |
| **V10.6** | Verify that all connections to external systems that involve sensitive information or functions are authenticated. |  | ✓ | ✓ |
| **V10.8** | Verify that there is a single standard TLS implementation that is used by the application that is configured to operate in an approved mode of operation. |  |  | ✓ |
| **V10.10** | Verify that TLS certificate public key pinning is implemented with production and backup public keys. For more information, please see the references below. |  |  | ✓ |
| **V10.11** | Verify that HTTP Strict Transport Security headers are included on all requests and for all subdomains, such as Strict-Transport-Security: max-age=15724800; includeSubdomains | ✓ | ✓ | ✓ |
| **V10.12** | Verify that production website URL has been submitted to preloaded list of Strict Transport Security domains maintained by web browser vendors. Please see the references below. |  |  | ✓ |
| **V10.13** | Ensure forward secrecy ciphers are in use to mitigate passive attackers recording traffic. | ✓ | ✓ | ✓ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **V10.14** | Verify that proper certification revocation, such as Online Certificate Status Protocol (OSCP) Stapling, is enabled and configured. | ✓ | ✓ | ✓ |
| **V10.15** | Verify that only strong algorithms, ciphers, and protocols are used, through all the certificate hierarchy, including root and intermediary certificates of your selected certifying authority. | ✓ | ✓ | ✓ |
| **V10.16** | Verify that the TLS settings are in line with current leading practice, particularly as common configurations, ciphers, and algorithms become insecure. | ✓ | ✓ | ✓ |

## HTTP security configuration verification requirements

Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **1** | **2** | **3** |
| **V11.1** | Verify that the application accepts only a defined set of required HTTP request methods, such as GET and POST are accepted, and unused methods (e.g. TRACE, PUT, and DELETE) are explicitly blocked. | ✓ | ✓ | ✓ |
| **V11.2** | Verify that every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8, ISO 8859-1). | ✓ | ✓ | ✓ |
| **V11.3** | Verify that HTTP headers added by a trusted proxy or SSO devices, such as a bearer token, are authenticated by the application. |  | ✓ | ✓ |
| **V11.4** | Verify that the Content Security Policy V2 (CSP) is in use for sites where content should not be viewed in a 3rd-party X-Frame. |  | ✓ | ✓ |
| **V11.5** | Verify that the HTTP headers or any part of the HTTP response do not expose detailed version information of system components. | ✓ | ✓ | ✓ |
| **V11.6** | Verify that all API responses contain X-Content-Type-Options: nosniff and Content-Disposition: attachment; filename="api.json" (or other appropriate filename for the content type). | ✓ | ✓ | ✓ |
| **v11.7** | Verify that the Content Security Policy V2 (CSP) is in use in a way that either disables inline JavaScript or provides an integrity check on inline JavaScript with CSP noncing or hashing. | ✓ | ✓ | ✓ |
| **v11.8** | Verify that the X-XSS-Protection: 1; mode=block header is in place. | ✓ | ✓ | ✓ |

## Malicious controls verification requirements

Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **1** | **2** | **3** |
| **V13.1** | Verify all malicious activity is adequately sandboxed, containerized or isolated to delay and deter attackers from attacking other applications. |  |  | ✓ |
| **V13.2** | Verify that a code review looks for malicious code, back doors, Easter eggs, and logic flaws. |  |  | ✓ |

## Business logic verification requirements

Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **1** | **2** | **3** |
| **V15.1** | Verify the application will only process business logic flows in sequential step order, with all steps being processed in realistic human time, and not process out of order, skipped steps, process steps from another user, or too quickly submitted transactions. |  | ✓ | ✓ |
| **V15.2** | Verify the application has business limits and correctly enforces on a per user basis, with configurable alerting and automated reactions to automated or unusual attack. |  | ✓ | ✓ |

## Files and resources verification requirements

Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **1** | **2** | **3** |
| **V16.1** | Verify that URL redirects and forwards only allow whitelisted destinations, or show a warning when redirecting to potentially untrusted content. | ✓ | ✓ | ✓ |
| **V16.2** | Verify that untrusted file data submitted to the application is not used directly with file I/O commands, particularly to protect against path traversal, local file include, file mime type, and OS command injection vulnerabilities. | ✓ | ✓ | ✓ |
| **V16.3** | Verify that files obtained from untrusted sources are validated to be of expected type and scanned by antivirus scanners to prevent upload of known malicious content. | ✓ | ✓ | ✓ |
| **V16.4** | Verify that untrusted data is not used within inclusion, class loader, or reflection capabilities to prevent remote/local file inclusion vulnerabilities. | ✓ | ✓ | ✓ |
| **V16.5** | Verify that untrusted data is not used within cross-domain resource sharing (CORS) to protect against arbitrary remote content. | ✓ | ✓ | ✓ |
| **V16.6** | Verify that files obtained from untrusted sources are stored outside the webroot, with limited permissions, preferably with strong validation. |  | ✓ | ✓ |
| **V16.7** | Verify that the web or application server is configured by default to deny access to remote resources or systems outside the web or application server. |  | ✓ | ✓ |
| **V16.8** | Verify the application code does not execute uploaded data obtained from untrusted sources. | ✓ | ✓ | ✓ |
| **V16.9** | Do not use Flash, Active-X, Silverlight, NACL, client-side Java or other client side technologies not supported natively via W3C browser standards. | ✓ | ✓ | ✓ |

## Mobile verification requirements

Requirement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **1** | **2** | **3** |
| **V17.1** | Verify that ID values stored on the device and retrievable by other applications, such as the UDID or IMEI number are not used as authentication tokens. | ✓ | ✓ | ✓ |
| **V17.2** | Verify that the mobile app does not store sensitive data onto potentially unencrypted shared resources on the device (e.g. SD card or shared folders). | ✓ | ✓ | ✓ |
| **V17.3** | Verify that sensitive data is not stored unprotected on the device, even in system protected areas such as key chains. | ✓ | ✓ | ✓ |
| **V17.4** | Verify that secret keys, API tokens, or passwords are dynamically generated in mobile applications. |  | ✓ | ✓ |
| **V17.5** | Verify that the mobile app prevents leaking of sensitive information (for example, screenshots are saved of the current application as the application is backgrounded or writing sensitive information in console) . |  | ✓ | ✓ |
| **V17.6** | Verify that the application is requesting minimal permissions for required functionality and resources. |  | ✓ | ✓ |
| **V17.7** | Verify that the application sensitive code is laid out unpredictably in memory (For example ASLR). | ✓ | ✓ | ✓ |
| **V17.8** | Verify that there are anti-debugging techniques present that are sufficient enough to deter or delay likely attackers from injecting debuggers into the mobile app (For example GDB). |  |  | ✓ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **V17.9** | Verify that the app does not export sensitive activities, intents, content providers etc., for other mobile apps on the same device to exploit. | ✓ | ✓ | ✓ |
| **V17.10** | Verify that mutable structures have been used for sensitive strings such as account numbers and are overwritten when not used. (Mitigate damage from memory analysis attacks). |  |  | ✓ |
| **V17.11** | Verify that the app’s exposed activities, intents, content providers etc. validate all inputs. | ✓ | ✓ | ✓ |

## Web services verification requirements

Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **1** | **2** | **3** |
| **v18.1** | Verify that the same encoding style is used between the client and the server. | ✓ | ✓ | ✓ |
| **v18.2** | Verify that access to administration and management functions within the Web Service Application is limited to web service administrators. | ✓ | ✓ | ✓ |
| **v18.3** | Verify that XML or JSON schema is in place and verified before accepting input. | ✓ | ✓ | ✓ |
| **v18.4** | Verify that all input is limited to an appropriate size limit. | ✓ | ✓ | ✓ |
| **v18.5** | Verify that SOAP based web services are compliant with Web Services-Interoperability (WS-I) Basic Profile at minimum. | ✓ | ✓ | ✓ |
| **v18.6** | Verify the use of session-based authentication and authorization. Please refer to sections 2, 3 and 4 for further guidance. Avoid the use of static "API keys" and similar. | ✓ | ✓ | ✓ |
| **v18.7** | Verify that the REST service is protected from Cross-Site Request Forgery. | ✓ | ✓ | ✓ |
| **v18.8** | Verify the REST service explicitly check the incoming Content-Type to be the expected one, such as application/xml or application/json. |  | ✓ | ✓ |
| **V18.9** | Verify that the message payload is signed to ensure reliable transport between client and service. |  | ✓ | ✓ |
| **V18.10** | Verify that alternative and less secure access paths do not exist. |  | ✓ | ✓ |

## Configuration

Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **1** | **2** | **3** |
| **v19.1** | All components should be up to date with proper security configuration(s) and version(s). This should include removal of unneeded configurations and folders such as sample applications, platform documentation, and default or example users. | ✓ | ✓ | ✓ |
| **v19.2** | Communications between components, such as between the application server and the database server, should be encrypted, particularly when the components are in different containers or on different systems. |  | ✓ | ✓ |
| **v19.3** | Communications between components, such as between the application server and the database server should be authenticated using an account with the least necessary privileges. |  | ✓ | ✓ |
| **v19.4** | Verify application deployments are adequately sandboxed, containerized or isolated to delay and deter attackers from attacking other applications. |  | ✓ | ✓ |
| **v19.5** | Verify that the application build and deployment processes are performed in a secure fashion. |  | ✓ | ✓ |
| **v19.6** | Verify that authorised administrators have the capability to verify the integrity of all security-relevant configurations to ensure that they have not been tampered with. |  |  | ✓ |
| **v19.7** | Verify that all application components are signed. |  |  | ✓ |
| **v19.8** | Verify that third party components come from trusted repositories. |  |  | ✓ |
| **v19.9** | Ensure that build processes for system level languages have all security flags enabled, such as ASLR, DEP, and security checks. |  |  | ✓ |

# Standard Compliance & Enforcement

## Compliance Measures

If applicable, compliance with the above Standard can be measured by the following criteria. Example evidence will vary depending on any supporting guidelines implemented to support this Standard. The following list is not exhaustive, and all example evidence types may not be required to validate compliance.

Evidence of compliance can be presented in hard copy or electronic format.

|  |  |
| --- | --- |
| **Criteria** | **Example Evidence** |
| Review some of the applications in use by the XXXX | * Confirm from the standard if the configuration settings on the applications are the same |

## Enforcement

All staff of XXXX must comply with all Information Security Standards. Failure to comply with these standards may result in disciplinary action in accordance with the current XXXX Human Resources policy. Disciplinary actions may include, but are not limited to:

* verbal and/or written warnings;
* instant dismissal; and
* actions by judicial and regulatory authorities.

# Exception Process / Glossary

## Exception Process

Non-compliance with the Standard statements described in this document must be reviewed and approved in accordance with the Exception Process defined in *XXXX-POL-ALL-001 - Information Security Policy Framework*

## Glossary / Acronyms

|  |  |
| --- | --- |
| ASLR | Address Space Layout Randomization |
| CSS | Cascading Style Sheets |
| CA | Certificate Authority |
| XSS | Cross-Site Scripting |
| DoS | Denial of Service |
| GUID | Globally Unique Identifier |
| HTML | Hyper Text Markup Language |
| HTTP | Hyper Text Transfer Protocol |
| LDAP | Lightweight Directory Access Protocol |
| PII | Personally Identifiable Information |
| TOV | Target of Verification |
| TLS | Transport Layer Security |
| URI | Uniform Resource Identifier |
| UAT | User acceptance testing |
| XML | Extended Markup Language |

# Document Management

## Document Revision Log

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Editor** | **Revision #** | **Description of Change** |
|  |  |  |  |

## Document Ownership

This Standard is owned by the YYYY

## Document Coordinator

This Standard is coordinated by the YYYY

## Document Approvers

|  |  |  |
| --- | --- | --- |
| **Approver Name** | **Signature** | **Date** |
|  |  |  |
|  |  |  |

## Distribution

* *Information Security*
* *IT Department*