This is a guidance box. Remove all guidance boxes after filling out the template. Items highlighted in turquoise should be edited appropriately. After all edits have been made, all highlights should be cleared.

Insert organization logo by clicking on the outlined image.

Secure Coding Standard Controls Template

Replace <organization name> with the name of the organization for the entire document. To do so, perform the following

- Press “Ctrl” + “H” keys simultaneously
- Enter “<organization name>” in the Find text box
- Enter your organization’s full name in the “Replace” text box
- Click “More”, and make sure “Match case” is ticked
- Click “Replace All”
- Close the dialog box.

Choose Classification

DATE: Click here to add date
VERSION: Click here to add text
REF: Click here to add text

Disclaimer

This template has been developed by the National Cybersecurity Authority (NCA) as an illustrative example that can be used by organizations as a reference and guide. This template must be customized and aligned with the <organization name>'s business and relevant legislative and regulatory requirements. This template must be approved by the head of the organization (Authorizing official) or his/her delegate. The NCA is not responsible for any use of this template as is, and it affirms that this template is solely an illustrative example.

Document Approval

Role	Job Title	Name	Date	Signature
Choose Role	<Insert job title>	<Insert individual's full personnel name>	Click here to add date	<Insert signature>

Version Control

Version	Date	Updated By	Version Details
<Insert version number>	Click here to add date	<Insert individual's full personnel name>	<Insert description of the version>

Review Table

Periodical Review Rate	Last Review Date	Upcoming Review Date
<Once a year>	Click here to add date	Click here to add date

Table of Contents

Purpose

This standard aims to define the detailed cybersecurity requirements related to coding protection for <organization name> in order to minimize cyber risks resulting from internal and external threats at <organization name>.

The requirements in this standard are aligned with the cybersecurity requirements issued by the National Cybersecurity Authority (NCA) including but not limited to (ECC-1:2018) and (CSCC-1:2019), in addition to other related cybersecurity legal and regulatory requirements.

Scope

This standard covers all software and application coding activities, projects and practices and information and technology assets at <organization name> and applies to all personnel (employees and contractors) at <organization name>.

Standards

1	Secure Code Development
Objective	To provide cybersecurity requirements to ensure the protection of software and application development activities and cybersecurity controls to secure developed software.
Risk Implication	Insecure code development practices could create security vulnerabilities that could be exploited to jeopardize the confidentiality, integrity and availability of <organization name>'s data and business operation.
Requirements	
1-1	A Secure Software Development Life Cycle (SSDLC) process Must be developed and implemented.
1-2	A DevSecOps methodology and process must be developed and adopted.

1-3	Cybersecurity requirements must be provided in the initial phases of software development and incorporated in the SSDLC process.
1-4	Cybersecurity testing must be conducted in the testing phases of software development and incorporated in the SSDLC process.
1-5	Automatic security testing must be developed and conducted in the testing phases of the SSDLC to detect malicious code injection, malware infection and known cybersecurity attacks.
1-6	Different techniques of cybersecurity testing must be used against all the development phases of the application (e.g. fuzzing, black box tests).
1-7	A secure environment must be designed and configured for development, testing and quality assurance purposes.
1-8	The secure coding guidelines under Table (A) must be implemented.
1-9	Memory-safe language must be used and the application must be tested against memory attacks.
1-10	Mitigations to the Open Web Application Security Project (OWASP) API Top 10 Application Security Risks must be implemented for critical systems and applications.
1-11	Mechanisms to restrict modification of production source code or production data must be implemented.
1-12	Third party vendors must be required to adhere to <organization name>'s cybersecurity policies and standard controls.
1-13	Only up-to-date, trusted and licensed sources of software development tools, libraries and components must be used.

1-14	Web application security controls must be implemented as per <organization name>'s Web Application Security Policy and Standard.
1-15	Standardized and extensively reviewed encryption algorithms must be used only as per relevant standard controls and procedures.
1-16	All versions of all software acquired from outside <organization name> must be verified to be still supported by the developer and appropriately hardened based on the developer's security recommendations.
1-17	Conduct training on writing secure code appropriate to the programming language and development environment being used for all software development personnel.
1-18	Ensure that all personnel involved in the SDLC of <organization name> are prepared to perform their SDLC-related roles and responsibilities throughout the SDLC.
1-19	<organization name> must secure and harden development endpoints (e.g., endpoints for software designers, developers, testers, etc.) by performing security development-related tasks using a risk-based approach.
1-20	<organization name> must limit access to development environment and enable event logs to ensure only authorized changes to code.
2	Source Code Repository
Objective	To provide cybersecurity controls to ensure the protection of source code, libraries, and source code repository.
Risk Implication	If the source code repository and libraries are not properly and sufficiently protected, <organization name>'s source code could be exposed, tampered or accessed without authorization.

Requirements	
2-1	A secure source code repository that has authentication, version control, and logging enabled must be used.
2-2	Deny access to source code and source code repository for anyone except application developers and owners when needed.
2-3	A unified version control numbering scheme must be used to reflect when updated versions of the software are installed.
2-4	Outdated versions of source code must be archived periodically.
2-5	Source code for applications under development must be segregated from source code for applications in production.
2-6	The source code of end of life applications must be archived to ensure that it can still be retrieved if needed.
2-7	A copy of the source code for all applications developed by third parties specifically for <organization name> must be acquired and stored in a secure source code repository.
2-8	Container and Docker security hardening standard controls and security best practices guidelines must be developed and implemented.
2-9	Secret management mechanisms must be deployed to manage secrets, keys and certifications and prevent storing secrets in containers.
2-10	Container images must be used from trusted or approved sources.
2-11	A private container registry must be used to ensure only verified and safe container images are downloaded to <organization name>'s system, and that every image is scanned for common known vulnerabilities.
2-12	Containers must not be run with super user accounts.

3	Secure Code Review and Testing
Objective	To provide assurance on the implemented secure coding cybersecurity controls and detect weaknesses, vulnerabilities and issues in software.
Risk Implication	If the source code and code development activities are not regularly tested and reviewed for security vulnerabilities, misconfigurations and weaknesses, <organization name> could be exposed to severe security risks.
Requirements	
3-1	A secure code review process must be conducted regularly for internally developed web applications.
3-2	Static and dynamic analysis tools must be applied to verify that secure coding practices are being adhered to for internally developed software.
3-3	Conduct a secure code review process regularly for all applications developed by third parties specifically for <organization name>.
3-4	Security controls of new internally developed applications must be reviewed and approved prior to application deployment into the production n environment.
3-5	Existing internally developed applications must be re-evaluated and re-approved after a significant change is made to the application, or after a predetermined period.
3-6	Risk assessments for all applications under development, or which are purchased, must be conducted to determine the controls required to mitigate application risks to acceptable limits prior to deployment into production environment (refer to <organization name>'s Risk Management Policy).
3-7	Cybersecurity compliance testing must be conducted for software against <organization name>'s cybersecurity policies

	and standard controls prior to deployment into production environment.
3-8	OWASP Application Security Verification Standard must be employed as a guide to define security requirements and generate test cases to review critical systems and applications.
3-9	Configurations review of software, including secure configuration hardening and patching, must be conducted prior to deployment into production environment.
3-10	Cybersecurity testing; including vulnerability assessment, penetrating testing and secure code review; must be conducted prior to deployment into production environment.
3-11	Cybersecurity testing, including vulnerability assessment and penetrating testing, must be conducted after deployment into production environment.
3-12	All developed application security issues discovered during the secure code review must be remediated prior to implementation into production environment.
3-13	Developed applications must be tested to ensure that Segregation of duties controls are appropriately implemented.
3-14	Test accounts and test data that are used in non-production environments must be removed before the application is moved into production.
3-15	Test and development environment must be logically separated from production and other environments using network restrictions by configuring Access-Control Lists (ACLs) and security policies on firewalls.
3-16	Source code peer-review must be conducted by a developer who did not write any of the code prior to its deployment into <organization name>'s production environment.
3-17	Only approved and licensed source code and software security assessment tools must be used.

3-18	Security testing for developed applications must be performed in all testing phases of SDLC including non-functional testing, Unit Testing (UT), System Integration Testing (SIT), and User Acceptance Testing (UAT).
3-19	A process and registry must be developed and maintained to manage software bugs, vulnerabilities and security issues.
3-20	Testing must be embedded as part of the Continuous Improvement/Continuous Development (CI/CD) pipeline.

Table A – Secure Coding Guidelines

A1	OWASP:A1:2021 Access Control – Broken Access Control
A1-1	It must be verified that users can only access secured functions or services for which they possess specific authorization.
A1-2	It must be verified that users can only access secured URLs for which they possess specific authorization.
A1-3	It must be verified that users can only access secured data files for which they possess specific authorization.
A1-4	It must verify that access controls are not bypassed.
A1-5	It must be verified that direct object references are protected in a way that ensures only authorized objects are accessible to each user.
A1-6	It must be verified that directory browsing is disabled unless required.
A1-7	It must be verified that users can only access protected data for which they possess specific authorization (for example, by implementing controls to protect against direct object reference tampering and prevent unauthorized access to data).
A1-8	It must be verified that access controls fail securely.
A1-9	It must be verified that the same access control rules implied by the presentation layer are enforced on the server side for that user role, and that controls and parameters cannot be re-enabled or re-added by users with higher privileges.
A1-10	It must be verified that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.
A1-11	It must be verified that all access controls are enforced on the server side.

A1-12	It must be verified that all access control decisions can be logged and all failed decisions are logged.
A1-13	It must be verified that the application or framework generates strong random anti-CSRF tokens unique to the user as part of all high value transactions or accessing protected data, and that the application verifies the presence of such tokens with the proper value for the current user when processing these requests.
A1-14	Aggregate access control protection – It must be verified that the system can protect against aggregate or continuous access of secured functions, resources, or data, possibly by the use of a resource governor, for example, to limit the number of registrations per hour or to prevent the entire database from being scraped by an individual user.
A1-15	It must be verified that a centralized mechanism (including libraries that call external authorization services) is in place to control access to each type of protected resource.
A1-16	It must be verified that there is segregation between code sections where privileges are elevated from other application code.
A1-17	Appropriate access controls must be implemented for protected data stored on the server. This includes cached data, temporary files and data accessible only by specific system users.
A1-18	It must be verified that service accounts or accounts supporting connections to or from external systems have the least privilege possible.
A1-19	It must be verified that account auditing is implemented and that unused accounts are disabled (for example, after more than 30 days from the expiration of an account's password, inactive account).
A1-20	If long authenticated sessions are allowed, a user's authorization must be periodically re-validated to ensure that

	their privileges have not changed. In case their privileges have changed, the user must be logged out and forced to re-authenticate (e.g., SMS, tokens, etc.)
A1-21	It must be verified that the application supports disabling of accounts and terminating sessions when authorization ceases (for example, upon changes to role, employment status, business process, etc.).
A2	OWASP:A2:2021 Cryptography – Cryptographic Failures
A2-1	It must be verified that all cryptographic functions used to protect secrets from the application user are implemented on the server side.
A2-2	It must be verified that all cryptographic modules fail securely.
A2-3	It must be verified that any master secret(s) is protected from unauthorized access (A master secret is an application credential stored as plaintext on disk that is used to protect access to security configuration information).
A2-4	It must be verified 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 unguessable by an attacker.
A2-5	It must be verified that cryptographic modules used by the application have been validated as per relevant policies and procedures.
A2-6	It must be verified that cryptographic modules operate in their approved mode in accordance with relevant policies and procedures.
A2-7	It must be verified that there is an explicit policy for how cryptographic keys are managed (for example, generated, distributed, revoked, or expired), and that this policy is properly enforced.

A2-8	It must be verified that non-repudiation through cryptography (digital signing) is present for sensitive transactions (e.g., financial or e-commerce transactions and record, etc.).
A2-9	It must be verified that all cryptographic keys are adequately protected. If a key has been compromised, it must no longer be trusted and must be replaced or revoked.
A2-10	It must be verified that Personally Identifiable Information (PII) and protected information and data are stored encrypted at rest.
A2-11	It must be verified that all forms containing protected information have disabled client-side caching, including autocomplete features.
A2-12	It must be verified that all protected data is sent to the server in the HTTP message body (i.e., URL parameters must never be used to send protected data).
A2-13	It must be verified that all cached or temporary copies of protected data stored on the server are protected from unauthorized access, and that those temporary working files are purged a soon as they are no longer required.
A2-14	Client-side caching or temporary copies of pages containing protected data must be disabled. Additionally, it must be verified that such copies are protected from unauthorized access or purged/invalidated after an authorized user accesses the protected data). (Cache-Control: no-store, may be used in conjunction with HTTP header control "Pragma: no-cache," which is less effective, but is HTTP/1.0 backward compatible).
A2-15	It must be verified that the list of protected data processed by the application is identified, and that there is an explicit policy for how access to this data must be controlled, and when this data must be encrypted (both at rest and in transit). Additionally, it must be verified that such policy is properly enforced.

A2-16	It must be verified that there is a method to remove each type of protected data from the application at the end of its required retention period.
A2-17	It must be verified that the application minimizes the number of parameters sent to untrusted systems, such as hidden fields, Ajax variables, cookies and header values.
A2-18	It must be verified that the application has the ability to detect and alert on abnormal numbers of requests for information, or on the processing of high value transactions for a user's role, such as screen scraping, automated use of web service extraction, or data loss prevention. For example, the average user must not be able to access more than 5 records per hour or 30 records per day.
A2-19	It must be verified that credentials used by the application on the server side; such as database connection, password and encryption secret keys; are not hard coded. Any credentials must be stored in a separate configuration file on a trusted system and must be encrypted.
A2-20	It must be verified that autocomplete features are disabled on forms expected to contain protected information, including authentication.
A3	OWASP:A3:2021 Input validation – Injection
A3-1	It must be verified that the runtime environment is not susceptible to buffer overflows, and that security controls prevent buffer overflows.
A3-2	It must be verified that the runtime environment is not susceptible to SQL Injection, and that security controls prevent SQL Injection.
A3-3	Access control methods for (SQL) must be used such as (LIMIT) to reduce the risk and the damage of the mass disclosure of information and records.

A3-4	It must be verified that the runtime environment is not susceptible to Cross Site Scripting (XSS), and that security controls prevent XSS.
A3-5	It must be verified that the runtime environment is not susceptible to LDAP Injection, and that security controls prevent LDAP Injection.
A3-6	It must be verified that the runtime environment is not susceptible to OS Command Injection, and that security controls prevent OS Command Injection.
A3-7	Data type, range and length must be verified (if possible).
A3-8	If any potentially hazardous characters must be allowed as input, additional controls; such as output encoding, secure task specific APIs, and accounting for the utilization of that data throughout the application; must be implemented. Examples of common hazardous characters include: ("\ '\\ + & ( ) % ' " < > :).
A3-9	It must be verified that all input validation is carried out by a centralized input validation routine for the application.
A3-10	It must be verified that all input validation failures result in input rejection or input sanitization.
A3-11	It must be verified that all input validation or encoding routines are performed and enforced on the server side.
A3-12	It must be verified that all untrusted data that is output to HTML (including HTML elements, HTML attributes, JavaScript data values, CSS blocks, and URL attributes) is properly discarded for the applicable context.
A3-13	It must be verified that a character set, such as UTF-8, is specified for all sources of input.
A3-14	It must be verified that all input data is cannibalized for all downstream decoders or interpreters prior to validation.
A3-15	If the application framework allows automatic mass parameter assignment (also called automatic variable binding) from the

	inbound request to a model, it must be verified that security sensitive fields such as “account Balance”, “role” or “password” are protected from malicious automatic binding.
A3-16	It must be verified 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.)
A3-17	It must be verified that a single input validation controls are used by the application for each type of data that is accepted.
A3-18	It must be verified that all input validation failures are logged.
A3-19	It must be verified that for each type of output encoding/escaping performed by the application, there is a single security control for that type of output for the intended destination.
A4	OWASP:A4:2021 Insecure Design
A4-1	Application processes and all high value business logic flows must be verified in a trusted environment, such as on a protected and monitored server.
A4-2	It must be verified that the application does not allow spoofed high value transactions, such as allowing Attacker User A to process a transaction as Victim User B, by tampering with or replaying session, transaction state, transaction or user IDs.
A4-3	It must be verified that the application does not allow high value business logic parameters to be tampered with, which include, but are not limited to, price, interest, discounts, PII, balances, stock IDs, etc.
A4-4	It must be verified that the application has defensive measures; such as verifiable and protected transaction logs, audit trails or system logs, and, in the highest value systems, real time monitoring of user activities and transactions for anomalies; to protect against repudiation attacks.

A4-5	It must be verified that the application protects against information disclosure attacks, such as direct object reference, tampering, session brute force or other attacks.
A4-6	It must be verified that the application has sufficient detection and governor controls to protect against brute force (such as the continuous use of a particular function) or denial of service attacks.
A4-7	It must be verified that the application has sufficient access controls to prevent elevation of privilege attacks. Such controls must include preventing anonymous users from accessing secured data or secured functions, and preventing users from accessing each other's details or using privileged functions.
A4-8	It must be verified that the application processes business logic flows in sequential steps only, with all steps being processed directly. Additionally, the application must be verified not to process out of order, skip steps, process steps from another user, or process transactions submitted quickly.
A4-9	It must be verified that 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.
A4-10	It must be verified that the application has business limits and enforces them in a trusted location (e.g., on a protected server) on a per user or per day basis, with configurable alerting and automated reactions to automated or unusual attack.
A4-11	Secure development lifecycle it must be established and used by AppSec professionals.
A4-12	Library of secure design patterns it must be established and used.
A4-13	Threat modeling it must be used for critical authentication, access control, business logic, and key flows.
A4-14	Security language and controls it must be integrated into user use cases.

A4-15	Security check it must be integrated at each tier of the application (from frontend to backend).
A4-16	Unit tests it must be written and tested to validate all critical flows are resistant against the threat model. It must compile use-cases *and* misuse-cases for each tier of the application.
A4-17	Depending on the exposure and protection needs tier layers must be segregated in the system and in the network layers,.
A4-18	Tenants must be segregated robustly by design throughout all tiers.
A4-19	The consumption of resource it must be limited by user or service.
A5	**OWASP:A5:2021 Communication Security – Security Misconfiguration**
A5-1	It must be verified 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.
A5-2	It must be verified that the latest version of TLS is used for all connections (including both external and backend connections) that are authenticated or involve protected data or functions.
A5-3	It must be verified that backend TLS connection failures are logged
A5-4	It must be verified that all connections to external systems that involve protected information or functions are authenticated
A5-5	It must be verified that all connections to external systems that involve protected information or functions use an account that has been set up to have the minimum privileges necessary for the application to function properly
A5-6	It must be verified that failed TLS connections do not fall back to an insecure connection
A5-7	It must be verified that certificate paths are built and verified for all client certificates using configured trust anchors and revocation information

A5-8	It must be verified that there is a single standard TLS implementation that is used by the application and configured to operate in an approved mode of operation
A5-9	It must be verified that specific character encodings are defined for all connections (e.g., UTF-8).
A5-10	Configuration must be verified and reviewed predictably in accordance with the most updated security configuration.
A5-11	It must be verified that the application accepts only a defined set of HTTP request methods, such as GET and POST, and that unused methods are explicitly blocked
A5-12	It must be verified that every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8)
A5-13	It must be verified that HTTP headers and/or other mechanisms for older browsers have been included to protect against click jacking attacks.
A5-14	It must be verified that HTTP headers in both requests and responses contain only printable ASCII characters
A5-15	The use of less complex data formats, such as JSON, must be verified, and serialization of protected data must be avoided
A5-16	All XML processors and libraries in use by the application or on the underlying operating system must be patched or upgraded. Additionally, dependency checkers must be used, and SOAP must be updated to SOAP 1.2 or higher
A5-17	XML external organization and DTD processing must be disabled in all XML parsers in the application, as per OWASP Cheat Sheet "XXE Prevention"
A5-18	Positive server-side input validation (whitelisting), filtering, or sanitization must be implemented to prevent hostile data within XML documents, headers, or nodes
A5-19	It must be verified that XML or XSL file upload functionality validates incoming XML using XSD validation or similar
A5-20	SAST and DAST tools must be used to help detect XXE in source code, although manual code review is the best

	alternative in large and complex applications with many integrations
A5-21	If the implementation of these controls is not possible, the use of virtual patching, API security gateways, or Web Application Firewalls (WAFs) must be considered to detect, monitor, and block XXE attacks
A5-22	It must be verified that parameterized queries are used to prevent SQL Injection
A5-23	It must be verified that strong and secure credentials are used for database access
A5-24	It must be verified that the application accessing the database uses the lowest possible level of privileges required.
A5-25	It must be verified that connection strings are not hard coded within the application, especially database authentication credentials
A5-26	It must be verified that the connection to the database is closed as soon as possible
A5-27	It must be verified that all unnecessary and unused database functionalities, including default vendor content, have been turned off or disabled. Only the minimum set of features and options required for the application to function must be installed. For example, unnecessary stored procedures or services and utility packages, must be disabled
A5-28	It must be verified that any default or unnecessary accounts with access to databases that are not required to support business requirements are disabled
A5-29	It must be verified that the application connects to the database with different credentials for every trust distinction and accountability (for example, user, read-only user, guest, administrators).
A5-30	It must be verified that remote logons and null sessions are disabled if not needed
A5-31	For applications that rely on a database, standard hardening configuration templates must be used, and all systems that are part of critical business processes must be tested

A6	OWASP:A6:2021 Malicious Code and Vulnerabilities – Vulnerable and Outdated Components
A6-1	It must be verified that no malicious code is in any code that was either developed or modified in order to create the application.
A6-2	It must be ensured that the integrity of interpreted code, libraries, executables, and configuration files is verified using checksums or hashes.
A6-3	It must be verified that all code implementing or using authentication controls is not affected by any malicious code.
A6-4	It must be verified that all code implementing or using session management controls is not affected by any malicious code.
A6-5	It must be verified that all code implementing or using access controls is not affected by any malicious code.
A6-6	It must be verified that all input validation controls are not affected by any malicious code.
A6-7	It must be verified that all code implementing or using output validation controls is not affected by any malicious code.
A6-8	It must be verified that all code supporting or using a cryptographic module is not affected by any malicious code.
A6-9	It must be verified that all code implementing or using error handling and logging controls is not affected by any malicious code.
A6-10	It must be verified all malicious activity is adequately sandboxed.
A6-11	Components must be updated with the latest patches as soon as a user knows about published vulnerabilities.
A6-12	Unused dependencies, unnecessary features, components, files, and documentation must be removed.

A6-13	Versions of both client-side and server-side components, (e.g., frameworks and libraries), and their dependencies must be continuously inventoried using tools such as versions, Dependency Check, retire.js, etc. Additionally, sources such as CVE and NVD, must be continuously monitored for vulnerabilities in the components, and software composition analysis tools must be used to automate the process. Subscription to email alerts for security vulnerabilities related to the used components must be ensured as well.
A6-14	Components must be obtained from official sources and over secure links only. Signed packages must be preferred to reduce the chance of including a modified, malicious component.
A6-15	Libraries and components that are unmaintained or do not create security patches for older versions must be monitored. If patching is not possible, deploying a virtual patch to monitor, detect, or protect against the discovered issue must be considered.
A6-16	It must be verified that URL redirects and forwards do not include invalidated data.
A6-17	It must be verified that filenames and path data obtained from untrusted sources are cannibalized to eliminate path traversal attacks.
A6-18	It must be verified that files obtained from untrusted sources are scanned by antivirus scanners to prevent the upload of known malicious content.
A6-19	It must be verified that parameters obtained from untrusted sources are not used in manipulating filenames, pathnames or any file system object without first being cannibalized and input validated to prevent local file inclusion attacks.
A6-20	It must be verified that parameters obtained from untrusted sources are cannibalized, input validated, and output encoded to prevent remote file inclusion attacks, particularly where input

	could be executed, such as header, source, or template inclusion.
A6-21	It must be verified that sharing remote IFRAMEs and HTML 5 resources across domains does not allow the inclusion of arbitrary remote content.
A6-22	It must be verified that files obtained from untrusted sources are stored outside the webroot.
A6-23	It must be verified that web or application server is configured by default to deny access to remote resources or systems outside the web or application server.
A6-24	It must be verified the application code does not execute uploaded data obtained from untrusted sources.
A6-25	It must be verified that Flash, Silverlight or other Rich Internet Application (RIA) cross-domain resource sharing configuration is set to prevent unauthenticated or unauthorized remote access.
A6-26	It must be verified that file types allowed for upload are limited to business purpose and needs only (e.g., PDF and office documents).
A6-27	It must be verified that file type validation is performed not only by checking file headers but also by checking file extension names.
A6-28	It must be verified that execution privileges are turned off on file upload directories.
A6-29	It must be verified that application files and resources are read-only by default.
A6-30	It must be verified that all unnecessary shares and administrative shares are removed, and that access to required shares is either restricted or requires authentication.
A6-31	Authentication must be required before allowing a file to be uploaded.

A6-32	Size of files that can be uploaded must be limited to the size that is needed for business purposes only (for example, maximum 1 MB), and a note must be added on the web page for the accepted file sizes.
A7	OWASP:A7:2021 Authentication Identification and Authentication Failures)
A7-1	It must be verified that all pages and resources require authentication except those specifically intended to be public (Principle of Complete Mediation).
A7-2	It must be verified that all password fields do not show users' passwords when entered, and that password fields (or the forms that contain them) have autocomplete disabled.
A7-3	It must be verified that all authentication controls fail securely to ensure that attackers cannot log in.
A7-4	It must be verified that credentials and all other ID organization information handled by the application do not traverse unencrypted or through weakly encrypted links.
A7-5	It must be verified that forgot password and other recovery paths do not send the existing or new passwords in clear text to the user.
A7-6	It must be verified that performing username enumeration is not possible via login, password reset, or forgot account functionalities.
A7-7	It must be verified that there are no default passwords in use for the application framework or any components used by the application (such as “admin/password”).
A7-8	It must be verified that a resource governor is in place to protect against vertical brute forcing (i.e., when a single account is tested against all possible passwords) and horizontal brute forcing (i.e., when all accounts are tested with the same password, such as “Password1”). A correct credential entry must incur no delay. For example, brute force source IP address lockout must be configured to 60 minutes and account

	lockout to 15 minutes. Both these governor mechanisms must be active simultaneously to protect against diagonal and distributed attacks.
A7-9	It must be verified that all authentication controls are enforced on the server side.
A7-10	It must be verified that password entry fields allow or encourage the use of passphrases, and do not prevent the entry of long passphrases or highly complex passwords, and provide a sufficient minimum strength to protect against the use of commonly chosen passwords.
A7-11	It must be verified that all account management functions (such as registration, update profile, forgot username, forgot password, disabled/lost token, help desk or IVR) that might regain access to the account are at least as resistant to attacks as the primary authentication mechanism.
A7-12	It must be verified that users can safely change their credentials using a mechanism that is at least as resistant to attacks as the primary authentication mechanism (e.g., SMS, tokens, mobile application, etc.). Password changes must require the existing password to be entered prior to entering a new password, followed by re-authentication of the user.
A7-13	It must be verified that authentication credentials expire after an administratively configurable period of time. The password expiry duration must be shorter based on the criticality of the application, thus ensuring a quicker password change.
A7-14	It must be verified that all authentication decisions are logged, including linear back offs and soft-locks.
A7-15	It must be verified that account passwords are salted using a salt that is unique to each account (e.g., internal user ID, account creation, etc.) and hashed before storing.
A7-16	It must be verified that all authentication credentials for accessing external services for the application are encrypted and stored in a protected location (not in source code).

A7-17	It must be verified that forgot password and other recovery paths send a time-limited activation token or use multi-factor authentication (e.g., SMS, tokens, mobile application, etc.) instead of a password.
A7-18	It must be verified that “forget password” functionality does not lock or otherwise disable the account until after the user has successfully changed their password.
A7-19	It must be verified that there are no shared knowledge questions/answers (Also called "secret" questions and answers).
A7-20	It must be verified that the system can be configured to disallow the use of a configurable number of previous passwords.
A7-21	It must be verified that all authentication controls (including libraries that call external authentication services) have a centralized implementation.
A7-22	It must be verified that re-authentication, step up or adaptive authentication, SMS or other two-factor application, or transaction signing is required before any application-specific sensitive operations are permitted as per the risk profile of the application.
A7-23	It must be verified that a functionality to invalidate or disable user credentials in the event of a compromise is in place.
A7-24	It must be verified that password encryption is implemented in accordance with relevant standard controls and procedures.
A7-25	If <organization name>’s application manages a credential store, it must ensure that only cryptographically strong one-way salted hashes of passwords are stored and that the table/file that stores the passwords and keys is write-able only by the application. (If possible, MD5 algorithm must not be used).
A7-26	Authentication logic must be segregated from the resource being requested, and redirection to and from the centralized authentication control must be used.

A7-27	Authentication failure responses must not indicate which part of the authentication data is incorrect. For example, instead of "Invalid username" or "Invalid password," "Invalid username and/or password" must be used for both. Error responses must be truly identical in both display and source code.
A7-28	It must be verified the strength of the password and check if it matches the most weak commonly used passwords.
A7-29	Password complexity and length requirements established by a policy or regulation approved by <organization name> must be enforced. Authentication credentials must be sufficient to withstand attacks that are typical of the threats in the deployed environment. Additionally, it must be verified that passwords contain: • At least 1 upper case character (A-Z) • At least 1 lower case character (a-z) • At least 1 digit (9-0) • At least 1 special character (e.g.,-,+* ()'&%$#"! “ ”~{|}`_^[\]@?<=>;:/) It must be verified that passwords do not contain: • More than 2 identical digits or characters in a row (e.g., 111, aa, etc.) • Sequential digits or characters (e.g., 123, 789, and abc) • The same username • Dictionary words (e.g., password, p@ssw0rd, secret123, etc.)
A7-30	Accounts must be disabled after an established number of invalid login attempts (e.g., five attempts for non-critical applications and three attempts for critical applications). Accounts must be disabled for a period of time sufficient to discourage brute force guessing of credentials, but not so long as to allow for a denial-of-service attack to be performed. (For example, disabled for 30 minutes).
A7-31	The last use (successful or unsuccessful) of a user account must be reported to the user at their next successful login.

A7-32	It must be verified that the framework's default session management control implementation is used by the application.
A7-33	It must be verified that sessions are invalidated when the user logs out.
A7-34	It must be verified that sessions timeout after a specified period of inactivity.
A7-35	It must be verified that all pages that require authentication to access them have logout links.
A7-36	It must be verified that the session ID is never disclosed other than in cookie headers, particularly in URLs, error messages, or logs. This includes verifying that the application does not support URL rewriting of session cookies.
A7-37	It must be verified that the session ID is changed or cleared on logout.
A7-38	It must be verified that authenticated session tokens using cookies are protected by the use of "Http Only" (cookies must not be displayed to the user).
A7-39	It must be verified that authenticated session tokens using cookies are protected with the "Secure" attribute and strict transport security headers (such as Strict-Transport-Security: max-age=60000; include Subdomains) is present.
A7-40	It must be verified that the session ID is changed on login to prevent session fixation.
A7-41	It must be verified that the session ID is changed on re-authentication.
A7-42	It must be verified that authenticated session tokens are sufficiently long and random to withstand attacks that are typical threats in the deployment environment.
A7-43	It must be verified that authenticated session tokens using cookies have their path set to an appropriately restrictive value

	for that site. The domain cookie attribute restriction must not be set except for a business requirement, such as a single sign on.
A7-44	It must be verified that the application does not permit duplicate concurrent user sessions, originating from different machines.
A7-45	It must be verified that sessions timeout after an administratively configurable maximum time period regardless of the performed activity (i.e., an absolute timeout).
A7-46	A new session identifier must be generated if the connection security is changed from HTTP to HTTPS, as can occur during authentication. Within an application, it is recommended to consistently utilize HTTPS rather than switching between HTTP to HTTPS.
A8	**OWASP:A8:2021 Insecure Deserialization – Software and Data Integrity Failures**
A8-1	Integrity checks, such as digital signatures, must be implemented on any serialized objects to prevent hostile object creation or data tampering.
A8-2	Strict type constraints during deserialization must be enforced before object creation as the code typically expects a definable set of classes. Bypasses to this technique have been demonstrated; therefore, reliance solely on this technique is not advisable.
A8-3	Code that reserializes must be isolated and run in low privilege environments whenever possible.
A8-4	Deserialization exceptions and failures; such as the cases in which the incoming type is not the expected type, or the deserialization throws exceptions; must be logged.
A8-5	Incoming and outgoing network connectivity from containers or servers that deserialize must be restricted or monitored.
A8-6	Deserialization must be monitored, and an alert must be issued if a user deserializes constantly.

A9	OWASP:A9:2021 Error Handling and Logging – Security logging and monitoring failures
A9-1	For in-house developed software, explicit error checking must be performed and documented for all input, including size, data type, and acceptable ranges or formats.
A9-2	It must be verified that the application does not output error messages or stack traces containing protected data that could assist an attacker, including a session ID and personal information.
A9-3	It must be verified that error handling is performed on trusted devices.
A9-4	It must be verified that all logging controls are implemented on the server.
A9-5	It must be verified that error handling logic in security controls denies access by default.
A9-6	It must be verified that security logging controls provide the ability to log both success and failure events that are identified as security-relevant.
A9-7	It must be verified that each log event includes a time stamp from a reliable source, severity level of the event, an indication that the event is a security relevant event (if mixed with other logs), the ID organization of the user that caused the event (if there is a user associated with the event), the source IP address of the request associated with the event, whether the event succeeded or failed, and a description of the event.
A9-8	It must be verified that all logs are protected from unauthorized access and modification.
A9-9	It must be verified that the application does not log application-specific protected data that could assist an attacker, including user's session IDs and personal or protected information.

A9-10	It must be verified that a log analysis tool is available which allows an analyst to search for log events based on a combination of search criteria across all fields in the log record format supported by this system.
A9-11	It must be verified that all events that include untrusted data will not execute as code in the intended log viewing software.
A9-12	It must be verified that there is a single logging implementation that is used by the application.
A9-13	It must be verified that logs have a standard regular procedure for backing up or archiving.
A9-14	“Try catch” must be implemented where applicable.
A9-15	It must be verified that all the below logs are enabled: • Log of all input validation failures • Log of all authentication attempts, especially failures • Log of all access control failures • Log of all apparent tampering events, including unexpected changes to data status. • Log of attempts to connect with invalid or expired session tokens • Log of all system exceptions • Log of all administrative functions, including changes to the security configuration settings • Log of all backend TLS connection failures • Log of cryptographic module failures
A10	OWASP:A10:2021 SSRF (Server-Side Request Forgery)
A10-1	Remote resource access functionality must be segmented in separate networks to reduce the impact of SSRF attack.
A10-2	<organization name> must enforce “deny by default” firewall policies or network access control rules to block all but essential intranet traffic.
A10-3	All client-supplied input data must be sanitized and validated

A10-4	It must enforce the URL schema, by specifying port, and destination against the whitelisting list.
A10-5	<organization name> must prohibit sending raw responses to client
A10-6	<organization name> must disable HTTP redirections
A10-7	<organization name> must ensure that personnel are aware of the URL consistency to avoid attacks such as DNS rebinding and “time of check, time of use” (TOCTOU) race conditions.
A10-8	It must not have other security relevant services installed on front systems (e.g. Open ID). Control local traffic on these systems (e.g. localhost)
A10-9	It must use network encryption (e.g. VPNs) on independent systems.
A11	Mobile Verification
A11-1	It must be verified that the client validates SSL certificates.
A11-2	It must be verified that Unique Device ID (UDID) values are not used as security controls.
A11-3	It must be verified that the mobile application does not store protected data on shared resources on a device (for example, on SD card or shared folders).
A11-4	It must be verified that protected data is not stored on SQLite database on the device.
A11-5	It must be verified that secret keys or passwords are not hard coded in the executable.
A11-6	It must be verified that the mobile application prevents the leakage of protected data via iOS auto snapshot feature.
A11-7	It must be verified that the application cannot be run on a jailbroken or rooted device.

A11-8	It must be verified that the session timeout is of a reasonable value.
A11-9	Requested permissions, as well as the resources authorized to be accessed (i.e., AndroidManifest.xml, iOS Entitlements), must be verified.
A11-10	It must be verified that crash logs do not contain protected data.
A11-11	It must be verified that the application binary has been obfuscated.
A11-12	It must be verified that all test data has been removed from the application container (.ipa .apk .bar).
A11-13	It must be verified that the application does not log protected data to the system log or file system.
A11-14	It must be verified that the application does not enable autocomplete for sensitive text input fields, such as password, personal information or credit card fields.
A11-15	It must be verified that the mobile application implements certificate pinning to prevent the proxying of application traffic.
A11-16	It must be verified that no misconfigurations are present in the configuration files (Debugging flags set, world readable/writable permissions).
A11-17	It must be verified that all third party libraries in use are up to date, and contain no known vulnerabilities.
A11-18	It must be verified that web data, such as HTTPS traffic, is not cached.
A11-19	It must be verified that the query string is not used for protected data. Instead, a POST request via SSL must be used with a CSRF token.
A11-20	It must be verified that, if applicable, any personal account numbers are truncated prior to storing them on a device.

A11-21	It must be verified that the application makes use of Address Space Layout Randomization (ASLR).
A11-22	It must be verified that data logged via the keyboard (iOS) does not contain credentials, financial information or other protected data.
A11-23	For Android applications, it must be verified that the application does not create files with permissions of MODE_WORLD_READABLE or MODE_WORLD_WRITABLE.
A11-24	It must be verified that protected data is stored in a cryptographically secure manner (even when stored on iOS keychain).
A11-25	It must be verified that anti-debugging and reverse engineering mechanisms are implemented in the application.
A11-26	It must be verified that the application does not export sensitive activities, intents, content providers, etc. on Android.
A11-27	It must be verified that mutable structures have been used for sensitive strings such as account numbers and are overwritten when not used, (to mitigate damage from memory analysis attacks).
A11-28	It must be verified that any exposed intents, content providers and broadcast receivers perform full data validation on input (Android).

Roles and Responsibilities

1- standard **Owner:** <head of the cybersecurity function>

2- standard **Review and Update:** <cybersecurity function>

3- standard **Implementation and Execution:** <information technology function>

4- standard **Compliance Measurement:** <cybersecurity function>

Update and Review

<cybersecurity function> must review the standard at least once a year or in case any changes happen to the infrastructure, policy, or the regulatory procedures in <organization name> or the relevant regulatory requirements.

Compliance

1- The <head of the cybersecurity function> will ensure compliance of <organization name> with this standard on a regular basis.

2- All personnel at <organization name> must comply with this standard.

3- Any violation of this standard may be subject to disciplinary action according to <organization name>'s procedures.