Eskom	**Standard**	

Title:	**Web Services Security Standard**	Document Identifier:	**240-107007584**
		Alternative Reference Number:	
		Area of Applicability:	**Eskom Holdings SOC Ltd**
		Functional Area:	**Group IT Security Services**
		Revision:	**3**
		Total Pages:	**16**
		Next Review Date:	**Jan 2024**
		Disclosure Classification:	**Controlled Disclosure**

Compiled by	**Functional Responsibility**	**Authorized by**
M. Ngidi	**C. Kungwane**	**S. Songo**
Senior Advisor – Information Security	**Middle Manager - Information Security**	**Senior Manager – IT Security Services**
Date: 27/03/2023	Date: 27 Match 2023	Date: 27-03-2023

Content

1. Introduction

Information within Eskom should be classified to indicate the need and to prioritise the level of protection required for the different systems. Eskom has defined five data classification levels namely: Public, Controlled disclosure, Confidential, Secret and Top Secret. This assessment aims to establish the level of trust that can be afforded to web applications security controls. The assessment results reveal the Actual Level of Trust (ALT) the web application can be afforded based on the implemented security controls deployed within that web application. The assessment further defines the Targeted Level of Trust (TLT) that each web application must achieve based on the system classification. The assessment can be used by application owners and application developers as:

- Baseline for required security control for their web application.
- Guideline for security controls that must be built within a web application.
- Input to security requirement when web applications are procured.

2. Supporting Clauses

2.1 Scope

This standard and the assessment applies to all systems residing on data networks, servers, mainframes and personal computers (stand-alone or network-enabled) located at Eskom and non-Eskom locations, where these systems are under the jurisdiction and/or ownership of Eskom, and any personal computers and/or servers authorised to access Eskom's data networks. The scope of this assessment also extends to computing equipment within the plant processing systems.

2.1.1 Purpose

The purpose of this document is to ensure that all web based systems/applications are assessed to ensure that appropriate security level is allocated to the application and gaps are identified and mitigated to ensure that all applications/systems are secured in harmony with the information classification levels.

2.1.2 Applicability

This standard applies to Eskom Holdings Limited, its divisions and subsidiaries, including temporary staff, contractors, service providers and consultants utilising Eskom's information resources.

2.2 Normative/Informative References

Parties using this document shall apply the most recent edition of the documents listed in the following paragraphs.

2.2.1 Normative

The following documents contain provisions that, through reference in the text, constitute requirements of this assessment. At the time of publication, the editions indicated were valid. Controlled documents are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent edition of the documents listed below. Information on currently valid national and international standards and specifications can be obtained from the Information Centre and Eskom Documentation Centre at Megawatt Park.

[1] 32-85 Information Security Policy

[2] 32-363 Asset and Information Security Classification Procedure

2.2.2 Informative

[1] ISO 27001 Information technology — Security techniques — Information security management systems — requirements

[2] The Open Web Application Security Project: Application Security Verification Standard 2013

[3] SANS Securing Web Application Technologies (i.e. SWAT)

2.3 Abbreviations

Abbreviation	Description
SANS	SysAdmin Audit Networking and Security Institute
OWASP	The Open Web Application Security Project
TLT	Targeted Level of Trust
ALT	Actual Level of Trust
SIEM	Security Incident and Event Management
OS	Operating System
IRSC	Information Risk Security and Compliance

2.4 Definitions

Term	Description
Username enumeration	The ability to compile a list of valid users based on a flaw in the registration process, login sequence, or password reset functionality that indicates the existence or not of a user based on input.
Master secret	A master secret is an application credential stored as plaintext on disk that is used to protect access to security configuration information
Authentication	Authentication is a process in which the credentials provided are compared to those on file in a database of authorized users' information on a local operating system or within an authentication server. If the credentials match, the process is completed and the user is granted authorization for

Term	Description
	access
Access control	Access control is a security technique that is used to regulate who or what can view or use resources in a computing environment. Access Control ensures that resources are only granted to those users who are entitled to them
Authorisation	Authorization is the approval, permission, or empowerment for someone or something to do something.

2.5 Monitoring Process

Internal Audit or the compliance & assurance section within Group IT will conduct annual assessments and audits to ascertain compliance with this standard. Quarterly reporting will be used internally within Group IT to track the effectiveness of the standard.

2.6 Related/Supporting Documents

240-86092116: Web Application Checklist

3. Assessment

The Web Application Security Standard is an adoption of the **Open Web Application Security Project's Application Security Verification Standard (OWASP-ASVS)** of 2013. This standard is conducted in a form of a checklist to ascertain the level of confidence in the security controls in place on the Web based applications. The checklist is divided into thirteen focus areas that are assessed in order to ascertain the security posture of the application/system.

The thirteen focus areas covered include:

- Authentication
- Session management
- Access control
- Input validation
- Cryptography,
- Error handling & logging,
- Data protection,
- Communication security,
- HTTP Security,
- Files & Resources,
- Mobile,
- Business logic
- Malicious controls.

Each focus area contains a set of controls and is assessed based on the answer to a number of questions addressing security requirements that must be met for the system/application to achieve the rating level. Each focus areas have a rating of four levels namely: 0, 1, 2, 3. Level zero implies that the system does not meet all the requirements to be rated as covering all requirements for level 1. If a system meets all requirements for level 1 it is then evaluated for the next level (i.e. Level 2) and so forth for the latter levels. The level of trust of the security control required per the level of data processed by the application is illustrated on *figure 1* below.

Eskom Data Classification	**Required Level of Trust**	**Security classification**
Public domain	0	Low
Controlled disclosure	1	Medium
Confidential	2	High
Secret	3	Very high

Figure 1

4. A Web Security Controls and Requirements

The controls and/ or requirements have been broken down into each focus area as indicated above.

4.1 Authentication

- All pages and resources require authentication except those intended for public consumptions.
- Where authentication is required, this needs to happen through an encrypted interface. Credentials and all other identity information handled by the application must not traverse in an unencrypted format or through weakly encrypted links or interfaces.
- Ensure that secure protocols are employed to assure the confidentiality and integrity of data
- It is prohibited for password fields to echo the user's password when it is entered. Password fields (or the forms that contain them) must have auto complete disabled.
- All authentication controls must fails securely to ensure attackers cannot log in unauthenticated.
- User credentials may not be hardcoded within the application
- Forgotten password and/or other recovery features must not send the existing or new passwords in clear text to the user.
- Username enumeration is not permitted via login, password reset, or forgot account functionality.

- Error messages on authentication must not reveal to the user the validity of the user such as stating that the entered password is incorrect which confirms to the attacker that the user id is valid. These should rather generically state that the username and/or password are incorrect, or supplied credentials are not valid.
- All authentication controls must be enforced from the server side.
- No default passwords may be used for the application framework or any components used by the application (such as "admin/password"). Where possible, well known administrator user accounts should be renamed.
- Where possible, a resource governor must be put in place to protect against vertical (a single account tested against all possible passwords) and horizontal brute forcing (all accounts tested with the same password e.g. "Password1").
    - A correct credential entry should incur no delay. For example, if an attacker tries to brute force all accounts with the single password "Password1", each incorrect attempt incurs a linear back off (say 5, 25, 125, 625 seconds) with a soft lock of say 15 minutes for that IP address before being allowed to proceed.
    - A similar control should also be in place to protect each account, with a linear back off configurable with a soft lock against the user account of say 15 minutes before being allowed to try again, regardless of source IP address.

  Both these governor mechanisms should be active simultaneously to protect against diagonal and distributed attacks.
- Password entry fields may allow or encourage the use of passphrases, and do not prevent long passphrases or highly complex passwords being entered, and provide a sufficient minimum strength to protect against the use of commonly chosen passwords.
- 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 attack as the primary authentication mechanism.
- Users must be able to safely change their credentials using a mechanism that is at least as resistant to attack as the primary authentication mechanism.
- Forgot password and other recovery paths should send a time-limited activation token or use two factor proofs (SMS, tokens, mobile application, etc.) rather than a password.
- Forgot password functionality should not lock or otherwise disable the account until after the user has successfully changed their password.
- The use of shared knowledge questions/answers (so called "secret" questions and answers) is prohibited as it is highly susceptible to social engineering threat.
- Authentication credentials should expire after an administratively configurable period of time. The configurable period must be aligned with the Logical Access Control standard.
- All authentication decisions must be logged, including linear back offs and soft-locks. Where applicable logs must be transferred to a SIEM solution
- Account passwords must be salted using a salt that is unique to that account (e.g., internal user ID, account creation) and hashed before storing.
- All authentication credentials for accessing services external to the application must be encrypted and stored in a protected location (not in the source code).

- The system must be configured to disallow the use of a configurable number of previous passwords. The minimum recommended previous password should be in line with the Eskom Logical Access Control standard.
- All authentication controls (including libraries that call external authentication services) must have a centralized implementation.
- 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 and or the transaction taking place. This is control is specifically required where data within the system has been classified as Secret or Top Secret.

4.2 Session Management

- The framework's default session management control implementation must be used by the application.
- Sessions must be invalidated when the user logs out.
- Sessions timeout after a specified period of inactivity must be implemented.
- All pages that require authentication to access them, must have logout links.
- Session id must never be 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.
- The session id must be changed or cleared on logout.
- Authenticated session tokens using cookies are protected by the use of "HttpOnly".
- 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.
- The session id must be changed on login to prevent session fixation.
- The session id must be changed on re-authentication.
- Only session ids generated by the application framework must be recognized as valid by the application.
- Authenticated session tokens must be sufficiently long and random to withstand attacks that are typical of the threats in the deployed environment.
- Authenticated session tokens using cookies must have their path set to an appropriately restrictive value for that site. The domain cookie attribute restriction should not be set unless for a business requirement, such as single sign on.
- The application must not permit duplicate concurrent user sessions, originating from different machines.
- Sessions timeout after an administratively-configurable maximum time period regardless of activity (an absolute timeout) must be implemented.

4.3 Access Control

- Users must only access secured functions or services for which they possess specific authorization.
- Users must only access secured URLs and or secured data files for which they possess specific authorization.
- Direct object references are protected, such that only authorized objects are accessible to each user.
- Directory browsing must be disabled unless deliberately desired.
- Users must only access protected data for which they possess specific authorization (for example, protect against direct object reference tampering).
- Access controls must fail securely.
- The same access control rules implied by the presentation layer must be enforced on the server side for that user role, such that controls and parameters cannot be re-enabled or re-added from higher privilege users.
- All user data attributes and policy information used by access controls must not be manipulated by end users unless specifically authorized.
- All access controls must be enforced on the server side.
- All access control decisions must be logged including all failed decisions.
- The application or framework must possess capability to generate strong random anti-CSRF tokens unique to the user as part of all high value transactions or accessing sensitive data, and that the application verifies the presence of this token with the proper value for the current user when processing these requests.
- Where applicable the system must have a capability to protect against aggregate or continuous access of secured functions, resources, or data. For example, possibly by the use of a resource governor to limit the number of registrations per hour or to prevent the entire database from being scraped by an individual user.
- Where applicable there must a centralized mechanism (including libraries that call external authorization services) for protecting access to each type of protected resource.

4.4 Input Validation

- The runtime environment must not be susceptible to buffer overflows, or that security controls must prevent buffer overflows.
- The runtime environment must not be susceptible to SQL Injection, or that security controls must prevent SQL Injection.
- The runtime environment must not be susceptible to Cross Site Scripting (XSS), or that security controls must prevent XSS.
- The runtime environment must not be susceptible to LDAP Injection, or that security controls must prevent LDAP Injection.

- The runtime environment must not be susceptible to OS Command Injection, or that security controls must prevent OS Command Injection.
- Input validation failures must result in input rejection or input sanitization.
- All input validation or encoding routines must be performed and enforced on the server side.
- All untrusted data that are output to HTML (including HTML elements, HTML attributes, JavaScript data values, CSS blocks, and URI attributes) must be properly escaped for the applicable context.
- A character set, such as UTF-8, must be specified for all sources of input. All input data must be canonicalized for all downstream decoders or interpreters prior to validation.
- If the application framework allows automatic mass parameter assignment (also called automatic variable binding) from the inbound request to a model – Security sensitive fields such as “accountBalance”, “role” or “password” must be protected from malicious automatic binding.
- Application must have defences 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.)
- A single input validation control must be used by the application for each type of data that is accepted.
- All input validation failures must be logged. For each type of output encoding/escaping performed by the application, there must be a single security control for that type of output for the intended destination.

4.5 Cryptography

- All cryptographic functions used to protect secrets from the application user must be implemented server side.
- All cryptographic modules must fail securely.
- Access to any master secret(s) must be 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).
- All random numbers, random file names, random GUIDs, and random strings must be generated using the cryptographic module’s approved random number generator when these random values are intended to be unguessable by an attacker.
- Cryptographic modules used by the application must be validated against FIPS 140-2 or an equivalent standard.
- Cryptographic modules must operate in their approved mode according to their published security policies.
- There must be a policy for how cryptographic keys are managed (e.g., generated, distributed, revoked, and expired).

4.6 Error Handling

- The application must not output error messages or stack traces containing sensitive data that could assist an attacker, including session id and personal information.
- All development language and framework generated errors must be suppressed or replaced by customised error messages as framework and development errors may reveal sensitive information to the user.
- All error handling must be performed on trusted devices.
- All logging controls must be implemented on the server.
- Error handling logic in security controls must deny access by default.
- Logging controls must provide the ability to log both success and failure events that are identified as security-relevant.
- Each log event must include: a time stamp from a reliable source, severity level of the event, an indication that this is a security relevant event (if mixed with other logs), the identity 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.
- Security logs must be protected from unauthorized access and modification.
- The application must not log application-specific sensitive data that could assist an attacker, including user's session ids and personal or sensitive information.
- A log analysis tool must be available as it will allow the analyst to search for log events based on combinations of search criteria across all fields in the log record format supported by this system.
- All events that include untrusted data must not execute as code in the intended log viewing software. There must be a single logging implementation that is used by the application.

4.7 Data Protection

- All forms containing sensitive information must have disabled client side caching, including autocomplete features.
- Ensure that secure protocols are employed to assure the confidentiality and integrity of data
- All sensitive data must be sent to the server in the HTTP message body (i.e. URL parameters must never be used to send sensitive data).
- All cached or temporary copies of sensitive data sent to the client must be protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data (e.g., the proper no-cache and no-store Cache-Control headers are set).
- All cached or temporary copies of sensitive data stored on the server must be protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data.
- The list of sensitive data processed by an application must be identified, and that there is an explicit policy for how access to the data must be controlled, and when the data must be encrypted (both at rest and in transit).

- There must be a method to remove each type of sensitive data from the application at the end of its required retention period.
- The application must minimize the number of parameters sent to untrusted systems, such as hidden fields, Ajax variables, cookies and header values.
- The application must have the ability to detect and generate alerts on abnormal numbers of requests for information or processing high value transactions for a user role, such as screen scraping, automated use of web service extraction, or data loss prevention. For example, the average user should not be able to access more than 5 records per hour or 30 records per day, or add 10 friends to a social network per minute.

4.8 Communication Security

- A path must be built from a trusted CA to each Transport Layer Security (TLS) server certificate, and that each server certificate must be valid.
- Latest TLS version must be used for all connections (including both external and backend connections) that are authenticated or that involve sensitive data or functions.
- Backend TLS connection failures must be logged.
- All connections to external systems that involve sensitive information or functions must be authenticated.
- All connections to external systems that involve sensitive information or functions must use an account that has been set up to have the minimum privileges necessary for the application to function properly.
- Failed TLS connections must not fall back to an insecure connection.
- Certificate paths must be built and verified for all client certificates using configured trust anchors and revocation information.
- There must be a single standard TLS implementation that is used by the application that is configured to operate in an approved mode of operation (See http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf).
- Specific character encodings must be defined for all connections (e.g., UTF-8).

4.9 HTTP Security

- The application must accept only a defined set of HTTP request methods, such as GET and POST and unused methods are explicitly blocked.
- Every HTTP response must contain a content type header specifying a safe character set (e.g., UTF-8).
- HTTP headers and / or other mechanisms for older browsers must be included to protect against click jacking attacks
- HTTP headers in both requests and responses must contain only printable ASCII characters.

4.10 File & Resources

- URL redirects and forwards must not include invalidated data.
- Filenames and path data obtained from untrusted sources must be canonicalized to eliminate path traversal attacks.
- Files obtained from untrusted sources must be scanned by anti-virus scanners to prevent upload of known malicious content.
- Parameters obtained from untrusted sources must not be used in manipulating filenames, pathnames or any file system object without first being canonicalized and input validated to prevent local file inclusion attacks.
- Parameters obtained from untrusted sources must be canonicalized, input validated, and output encoded to prevent remote file inclusion attacks, particularly where input could be executed, such as header, source, or template inclusion
- Remote IFRAMEs and HTML 5 cross-domain resource sharing must not allow inclusion of arbitrary remote content.
- Files obtained from untrusted sources must be stored outside the ebroot.
- Web or application server must be configured by default to deny access to remote resources or systems outside the web or application server.
- Application code must not execute uploaded data obtained from untrusted sources.
- Flash, Silverlight or other rich internet application (RIA) cross domain resource sharing configuration must be configured to prevent unauthenticated or unauthorized remote access.

4.11 Malicious Control

- No malicious code must be in any code that was either developed or modified in order to create the application.
- The integrity of interpreted code, libraries, executables and configuration files must be verified using checksums or hashes.
- All code implementing or using authentication controls must not be affected by any malicious code.
- All code implementing or using session management controls must not be affected by any malicious code.
- All code implementing or using access controls must not be affected by any malicious code.
- All input validation controls must not be affected by any malicious code.
- All code implementing or using output validation controls must not be affected by any malicious code.
- All code supporting or using a cryptographic module must not be affected by any malicious code.
- All code implementing or using error handling and logging controls not be affected by any malicious code.

- All malicious activity must be adequately sandboxed.
- Sensitive data must be rapidly sanitized from memory as soon as it is no longer needed.

4.12 Business Logic

- All application processes or high value business logic must flow in a trusted environment, such as a protected and monitored server.
- The application must 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.
- The application must not allow high value business logic parameters to be tampered with, such as (but not limited to): price, interest, discounts, personal identifiable information, balances, stock IDs, etc.
- The application must have defensive measures to protect against repudiation attacks, such as verifiable and protected transaction logs, audit trails or system logs, and in highest value systems real time monitoring of user activities and transactions for anomalies.
- The application must protect against information disclosure attacks, such as direct object reference, tampering, session brute force or other attacks.
- The application must have sufficient detection and governing controls to protect against brute force (such as continuously using a particular function) or denial of service attacks.
- The application must have sufficient access control to prevent elevation of privilege attacks, such as allowing anonymous users from accessing secured data or secured functions, or allowing users to access each other’s details or using privileged functions.
- The application must 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.
- The application must have 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.
- The application must have business limits and enforces them in a trusted location (as on a protected server) on a per user, per day or daily basis, with configurable alerting and automated reactions to automated or unusual attack. Examples include (but not limited to): ensuring new SIM users don’t exceed R1000 per day for a new phone account, a forum allowing more than 100 new users per day or preventing posts or private messages until the account has been verified, a health system should not allow a single doctor to access more patient records than they can reasonably treat in a day, or a small business finance system allowing more than 200 000 invoice payments or R1 000 000 000 0000 per day across all users.
- In all cases, the business limits and totals should be reasonable for the business concerned. The only unreasonable outcome is if there are no business limits, alerting or enforcement.

4.13 Mobile

- The client must validate SSL certificates
- Unique device ID (UDID) values must not be used as security controls.
- The mobile app must not store sensitive data onto shared resources on the device (e.g. SD card or shared folders)
- Sensitive data must not be stored in a database on the device (e.g. SQLite).
- Secret keys or passwords must not be hard-coded in the executable.
- The mobile app must prevent leaking of sensitive data via auto snapshot feature of iOS.
- The app must not run on a jail broken or rooted device.
- The session timeout must be of a reasonable value.
- The application must verify the permissions being requested as well as the resources that are authorized to access (i.e. AndroidManifest.xml, iOS Entitlements).
- Crash logs must not contain sensitive data.
- The application binary must be obfuscated.
- All test data must be removed from the app container (.ipa, .apk, .bar).
- The application must not log sensitive data to the system log or filesystem.
- The application must not enable autocomplete for sensitive text input fields, such as passwords, personal information or credit cards.
- The mobile app must implement certificate pinning to prevent the proxying of app traffic.
- No misconfigurations must be present in the configuration files (Debugging flags set, world readable/writable permissions).
- Any 3rd-party libraries in use must be up to date, must not contain any known vulnerabilities.
- Web data, such as HTTPS traffic, must not be cached.
- The query string is not used for sensitive data. Instead, a POST request via SSL should be used with a CSRF token.
- If applicable, any personal account numbers must be truncated prior to storing on the device.
- The application must make use of Address Space Layout Randomization (ASLR).
- Data logged via the keyboard (iOS) must not contain credentials, financial information or other sensitive data.
- An Android app, must verify that the app does not create files with permissions of MODE_WORLD_READABLE or MODE_WORLD_WRITABLE
- Sensitive data must be stored in a cryptographically secure manner (even when stored in the iOS keychain).
- Anti-debugging and reverse engineering mechanisms must be implemented in the app.
- The app must not export sensitive activities, intents, content providers etc. on Android.

- Mutable structures must be used for sensitive strings such as account numbers and are overwritten when not used. (Mitigate damage from memory analysis attacks).
- Any exposed intents, content providers and broadcast receivers must perform full data validation on input (Android).

5. Acceptance

This document has been seen and accepted by:

Name	Designation
Faith Burn	Chief Information Officer
Sithembile Songo	Senior Manager – IT Security Services
Tebogo Makhwelo	Senior Manager – Infrastructure Operations
Ian Marks	Senior Manager – Specialised Technical Services (Acting)
Anthenia Phuku	Senior Manager – IM Business Solutions and Development Services
Varsha Pillay	Senior Manager – Applications Operations
Ian Marks	Senior Manager – IT Governance Services (Acting)
Grasswell Mabudusha	Senior Manager – Strategic Project Services

6. Revisions

Date	Rev.	Compiler	Remarks
July 2015	0.1	Xolani Lukhele	Initial Draft
October 2015	0.2	Xolani Lukhele	Update document with input
April 2016	1	Xolani Lukhele	Update document with input
October 2019	2	Neo Lemao	Update document with input
November 2022	3	Neo Lemao	Update document with input
March 2023	4	Mabongi Ngidi	Document reviewed

7. Development Team

The following people were involved in the development of this document:

- Neo Lemao (SME)
- Mmabatho Singo