NATIONAL DATA
MANAGEMENT AUTHORITY

Secure Coding Standard

Prepared By:

National Data Management Authority
March 2023

Document Status Sheet

	Signature	Date
Policy Coordinator (Cybersecurity)	**Muriana McPherson**	**31-03-2023**
General Manager (NDMA)	**Christopher Deen**	**31-03-2023**

Document History and Version Control

Date	Version	Description	Authorised By	Approved By
31-03-2023	**1.0**		**General Manager, NDMA**	**National ICT Advisor**

Summary

1. This standard addresses secure coding practices for software development.
2. It was adapted from NIST Cybersecurity Framework Policy Template Guide and SANS Institute.
3. This is a living document which will be updated annually or as required.
4. Submit all inquiries and requests for future enhancements to the Policy Coordinator, NDMA

1.0 Purpose

There has been an increase in the number and frequency of cyber-attacks that attempt to exploit vulnerabilities within computer systems and thereby threaten the confidentiality, integrity, and availability of information. Many vulnerabilities that are successfully exploited are due to software coding weaknesses and coding implementation flaws.

The objective of this coding standard is to ensure that code written is resilient to growing threats and to avoid the occurrence of the most common coding errors which create serious vulnerabilities in software. While it is impossible to write code that is completely impervious to all possible attacks, implementing these coding standards throughout information systems will significantly reduce the risk of disclosure, alteration, or destruction of information due to software vulnerabilities.

2.0 Authority

The Permanent Secretary, Administrative Head, Head of Human Resources or their designated representative of the Public Sector Organisation is responsible for the implementation of this standard. For further information regarding the foregoing, please contact the Policy Coordinator - National Data Management Authority (NDMA).

3.0 Scope

This standard encompasses all systems, automated and manual, for which the Government of Guyana has administrative responsibility, including systems managed or hosted by third parties on behalf of the Government. It addresses all information, regardless of the form or format, which is created or used in support of business activities. It is the user's responsibility to read and understand this standard and to conduct their activities in accordance with its terms.

4.0 Standard

As per the Information Security Policy, all software written for or deployed on systems must incorporate secure coding practices, to avoid the occurrence of common coding vulnerabilities and to be resilient to high-risk threats, before being deployed in production.

The items enumerated in this standard are not an exhaustive list of high-risk attacks and common coding errors but rather a list of the most damaging and pervasive. Therefore, code written must contain mitigating controls not only for the items specifically articulated in the standard below, but also for any medium and high-risk threats that are identified during a system's life cycle.

High risk threats include, but are not limited to:

4.1 Code Injection

4.2 Cross-site scripting (XSS)

4.3 Cross-site request forgery (CSRF)

4.4 Information leakage and improper error handling

4.5 Missing Authentication for Critical Function

4.6 Missing Encryption of Sensitive Data

4.7 URL Redirection to Untrusted Site ('Open Redirect')

At a minimum, code must eliminate or mitigate the threats identified in the current version of the Open Web Application Security Project (OWASP) Top 10 Most Critical Application Security Risks ('OWASP Top 10') [1]and the Common Weakness Enumeration (CWE)/SANS Top 25 Most Dangerous Software Errors ('CWE/SANS Top 25')[2] publications (see Appendix A).

Both OWASP and CWE/SANS periodically reissue their respective lists based on changes in vulnerability and exploitation patterns. Developers are required to independently remain aware of updates to these lists and incorporate any new recommendations.

Use of common security control libraries and common API's, that have undergone security testing, are required to ensure a consistent approach that minimises defects and prevents exploitation. When available, publicly available or vendor-supplied libraries or APIs should be used unless there's a business case developed and exception granted by the Information Security Officer (ISO)/designated security representative to develop a custom library.

To prevent defects or detect and remove them early, code must be checked for errors throughout development and during maintenance. This process can help to realise significant cost and schedule benefits to the organisation.

Organisations must verify that the software assurance model used by the vendor is in line with this standard through vendor assurances, security testing and/or contract requirements.

5.0 Compliance

This standard shall take effect upon publication. Compliance is expected with all organisational policies and standards. Failure to comply with the standard may, at the full discretion of the Permanent Secretary, Administrative Head, or Head of Human Resources of the Public Sector Organisation, may result in the suspension of any or all privileges and further action may be taken by the Ministry of Public Service.

6.0 Exceptions

Requests for exceptions to this standard shall be reviewed by the Permanent Secretary, Administrative Head, Head of Human Resources of the Public Sector Organisation, or the Policy Coordinator, NDMA. Departments requesting exceptions shall provide written requests to the relevant personnel. The request should specifically state the scope of the exception along with justification for granting the exception, the potential impact or risk attendant upon granting the exception, risk mitigation measures to be undertaken by the IT Department, initiatives, actions and a time-frame for achieving the minimum compliance level with the policies set forth herein.

---

[1] *Retrieved from*: OWASP Top Ten https://owasp.org/www-project-top-ten/

[2] *Retrieved from* Mitre's Common Weakness Enumeration Top 25 list http://cwe.mitre.org/top25/archive/2021/2021_cwe_top25.html

7.0 Maintenance

The Policy Coordinator, NDMA shall be responsible for the maintenance of this standard.

8.0 Definitions of Key Terms

Term	Definition
Authentication[3]	Verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system.
Code[4]	System of communication in which arbitrary groups of letters, numbers, or symbols represent units of plain text of varying length.
Cross-site Request Forgery (CSRF)[5]	An attack in which a subscriber currently authenticated to an RP and connected through a secure session browses to an attacker's website, causing the subscriber to unknowingly invoke unwanted actions at the RP. For example, if a bank website is vulnerable to a CSRF attack, it may be possible for a subscriber to unintentionally authorize a large money transfer, merely by viewing a malicious link in a webmail message while a connection to the bank is open in another browser window.
Cross-site Scripting (XSS)[6]	A vulnerability that allows attackers to inject malicious code into an otherwise benign website. These scripts acquire the permissions of scripts generated by the target website and can therefore compromise the confidentiality and integrity of data transfers between the website and client. Websites are vulnerable if they display user-supplied data from requests or forms without sanitizing the data so that it is not executable.
Encryption[7]	Cryptographic transformation of data (called "plaintext") into a form (called "ciphertext") that conceals the data's original meaning to prevent it from being known or used.

[3] *Retrieved from:* NIST Information Technology Laboratory Computer Security Resource Center https://csrc.nist.gov/glossary/term/authentication

[4] *Retrieved from:* NIST Information Technology Laboratory Computer Security Resource Center https://csrc.nist.gov/glossary/term/code

[5] *Retrieved from:* NIST Information Technology Laboratory Computer Security Resource Center https://csrc.nist.gov/glossary/term/cross_site_request_forgery

[6] *Retrieved from:* NIST Information Technology Laboratory Computer Security Resource Center https://csrc.nist.gov/glossary/term/cross_site_scripting

[7] *Retrieved from:* NIST Information Technology Laboratory Computer Security Resource Center https://csrc.nist.gov/glossary/term/encryption

Term	Definition
Injection[8]	Attacks that look for web sites that pass insufficiently-processed user input to database back-ends
URL Uniform Resource Locator[9]	Reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it. A typical URL could have the form http://www.example.com/index.html, which indicates a protocol (http), a host name (www.example.com), and a file name (index.html). Also sometimes referred to as a web address.
Vulnerability[10]	Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source.

9.0 Contact Information

Submit all inquiries and requests for future enhancements to the Policy Coordinator, NDMA.

10.0 Related Documents

Open Web Application Security Project (OWASP) Top 10 Most Critical Application Security Risks ('OWASP Top 10') [11]

Open Web Application Security Project (OWASP) Developer Cheat Sheets [12]

Common Weakness Enumeration (CWE)/SANS Top 25 Most Dangerous Software Errors 'CWE/SANS Top 25') [13]

Common Weakness Enumeration (CWE) List [14]

Carnegie Mellon Software Engineering Institute CERT Secure Coding Standards [15]

Appendix A: Coding Resources

[8] *Retrieved from:* NIST Information Technology Laboratory Computer Security Resource Center https://csrc.nist.gov/glossary/term/sql_injection

[9] *Retrieved from:* NIST Information Technology Laboratory Computer Security Resource Center https://csrc.nist.gov/glossary/term/uniform_resource_locator

[10] *Retrieved from:* NIST Information Technology Laboratory Computer Security Resource Center https://csrc.nist.gov/glossary/term/vulnerability

[11] *Retrieved from*: OWASP Top 10: https://owasp.org/www-project-top-ten/

[12] *Retrieved from*: OWASP Cheat Sheet Series https://cheatsheetseries.owasp.org/

[13] *Retrieved from* Mitre's Common Weakness Enumeration Top 25 Most Dangerous Software Weaknesses http://cwe.mitre.org/top25/archive/2021/2021_cwe_top25.html

[14] *Retrieved from*: Mitre's Common Weakness Enumeration List http://cwe.mitre.org/data/index.html

[15] *Retrieved from*: Carnegie Mellon CERT https://wiki.sei.cmu.edu/confluence/display/seccode

Open Web Application Security Project (OWASP)

The OWASP Top 10 is authored by OWASP, an open-source application security community project which aims to raise security awareness of web application security risks. Although OWASP is focused on web application security, the standards and controls presented by this organisation are generally also applicable to non-web-based information systems.

In addition to the "Top 10" list, OWASP also produces the Enterprise Security API (ESAPI) library[16] and developer cheat sheets[17]. The ESAPI library is an open source, web application security control library designed to mitigate risks to web applications. The ESAPI library provides a framework to implement code to address the risks listed within the OWASP Top Ten project. The cheat sheets provide a concise collection of high value information on specific web application security topics.

Common Weakness Enumeration/SANS

The CWE/SANS Top 25 Most Dangerous Software Errors publication is the result of collaboration between the SANS Institute, MITRE, and many top software security experts in the US and Europe. The publication is a list of the most widespread and critical errors that can lead to serious vulnerabilities in software. They are often easy to find, and easy to exploit. They are dangerous because they will frequently allow attackers to completely take over the software, steal data, or prevent the software from working at all.

The MITRE website provides detailed guidance to software programmers for mitigating and avoiding each of the common weaknesses enumerated within the Top 25 list with the Common Weakness Enumeration (CWE) List. [18]

[16] *Retrieved from*: OWASP Enterprise Security API https://owasp.org/www-project-enterprise-security-api/

[17] *Retrieved from*: OWASP Cheat Sheet Series https://cheatsheetseries.owasp.org/

[18] *Retrieved from*: Mitre's Common Weakness Enumeration List http://cwe.mitre.org/data/index.html