NEW YORK STATE OF OPPORTUNITY. Office of Information Technology Services

State Capitol P.O. Box 2062
Albany, NY 12220-0062
www.its.ny.gov

New York State Information Technology Standard	**No:** NYS-S13-002
IT Standard: **Secure Coding**	**Updated:** 5/20/2021
	Issued By: NYS Office of Information Technology Services **Owner:** Chief Information Security Office

1.0 Purpose and Benefits

Government organizations are under constant 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 for New York State is resilient to high-risk 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 the State's information systems will significantly reduce the risk of disclosure, alteration, or destruction of the State's information due to software vulnerabilities.

2.0 Authority

Section 103(10) of the State Technology Law provides the Office of Information Technology Services (ITS) with the authority to establish statewide technology policies, including technology and security standards. *Section 2* of *Executive Order No. 117*, established January 2002, provides the State Chief Information Officer with the authority to oversee, direct and coordinate the establishment of information technology policies, protocols, and standards for State government, including hardware, software, security, and business re-engineering. Details regarding this authority can be found in NYS ITS Policy, *NYS-P08-002 Authority to Establish Enterprise Information Technology (IT) Policies, Standards and Guidelines*.

3.0 Scope

This standard applies to all “State Entities” (SE), defined as “State Government” entities as defined in Executive Order 117, established January 2002, or “State Agencies” as defined in Section 101 of the State Technology Law. This includes employees and all third parties (such as local governments, consultants, vendors, and contractors) that use or access any ITS resource for which ITS the SE has administrative responsibility, including systems managed or hosted by third parties on behalf of the ITS SE. While an SE may adopt a different standard, it must include the requirements set forth in this one.

This standard covers all systems and software developed for the SE, including but not limited to mobile, web, desktop, database, or mainframe regardless of their current system life cycle phase. This includes all test, quality control, production, and other ad-hoc systems that exist within or external to the State network. This standard equally applies to SE systems that are developed by third-party entities with or without the participation of State development staff. It is the responsibility of the SE to ensure vendor supplied software, including custom or commercial off-the-shelf (COTS), is built securely.

4.0 Information Statement

As per the NYS-P03-002 NYS Information Security Policy, all software written for or deployed on SE 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.

4.1 Secure Coding Practices

The development process must use secure coding practices and follow the NYS-S13-001 Secure System Development Life Cycle Standard.

These secure coding practices can include, but are not limited to the following list:

- **Identify security requirements upfront** in the development life cycle and make sure that subsequent development artifacts are evaluated for compliance with those requirements.
- **Anticipate threats** to which the software will be subjected and then develop threat mitigation strategies that address those threats.
- **Validate input data** from all data sources including, but not limited to, command line arguments, network interfaces, environmental variables, and user-controlled files. Proper input validation can eliminate most software vulnerabilities.
- **Validate/limit output data** sent to other systems such as command shells, relational databases, and COTS components to prevent misusing functionality in such systems.

- **Keep the code simple** as complexity will increase the likelihood that errors will be introduced.
- **Use industry best practices for authentication methods and services** to validate users accessing the system and all resources within the system.
- **All developed code must be stored** in a secure repository.
- **Whenever possible, use tested and approved code for common tasks** rather than create new, untested code.
- **Restrict error handling** from providing details of how the application works or about the system upon which it resides.
- **Generate log files** per system requirements and the NYS-S14-005 Security Logging Standard. These log files would be relevant to forensic analysis in the event of an incident.
- **Perform code scanning** per NYS-S15-002 Vulnerability Management Standard on both SE developed code, any open source components utilized by the SE, and any third-party developed code.
- **Use peer reviews** by team members who can draw upon their respective knowledge and experience to uncover potential issues.
- **Document/comment the code** for easier maintenance and remediation of any security issues.
- **Remain aware of current threats and vulnerabilities** to the code and respective technologies in use by the SE**.** Please see Exhibit 1 for some examples.

4.2 Security Risk Assessments

Security risk assessments are required for use of current frameworks, common security control libraries, and common application programming interfaces (APIs). This provides a consistent approach that minimizes defects and prevents vulnerabilities from being incorporated into SE code.

Code must be checked for errors throughout development and during maintenance in order to prevent defects, or detect and remove them early, often realizing cost and schedule benefits to the SE.

4.3 Vulnerability Scanning

The NYS-S15-002 Vulnerability Management Standard requires that systems undergo source code analysis before moving between environments if there has been a change to application code. As per NYS-P03-002 Information Security Policy and NYS-S15-002 Vulnerability Management Standard appropriate action must be taken to address discovered vulnerabilities.

This process is often done by:

- **Peer Review** – programmers with extensive knowledge and experience who did not author the code review the code to ensure it follows established secure coding principles and best practices.
- **Automated Testing** – automated programs can be used to check the security of an application using static and dynamic analysis tools.
    - Static analysis tools – analyze the source code of an application without executing the application. Static code scanning is done throughout the development process as well as prior to being moved into the production environment.
    - Dynamic analysis tools – analyze the application during execution in a runtime environment.

When software development is outsourced, SEs must verify that the software assurance model used by the vendor is in line with this standard through SE security testing and/or contract requirements.

5.0 Compliance

This standard shall take effect upon publication. Compliance is required with all enterprise policies and standards. ITS may amend its policies and standards at any time; compliance with amended policies and standards is required.

If compliance with this standard is not feasible or technically possible, or if deviation from this standard is necessary to support a business function, SEs shall request an exception through the Chief Information Security Office (CISO) exception process.

6.0 Definitions of Key Terms

Except for terms defined in this standard, all terms shall have the meanings found in http://www.its.ny.gov/glossary.

7.0 Contact Information

Submit all inquiries and requests for future enhancements to the standard owner at:

Chief Information Security Office
Reference: NYS-S13-002
NYS Office of Information Technology Services
1220 Washington Avenue, Building 5
Albany, NY 12226
Telephone: (518) 242-5200
Email: CISO@its.ny.gov

Statewide technology policies, standards, and guidelines may be found at the following website: http://www.its.ny.gov/tables/technologypolicyindex

8.0 Revision History

This standard should be reviewed consistent with the requirements set forth in NYS-P08-002 Authority to Establish State Enterprise Information Technology (IT) Policy, Standards and Guidelines.

Date	Description of Change	Reviewer
10/18/2013	Original Standard Release	Thomas Smith, Chief Information Security Officer
10/17/2014	Added reference to NYS Information Security Policy, technical correction of "cluster ISO" reference to "ISO/designated security representative"	Deborah A. Snyder, Acting Chief Information Security Officer
03/10/2017	Updated Scope, contact information and rebranded.	Deborah A. Snyder, Deputy Chief Information Security Officer
09/11/2018	Scheduled review – minor changes to Authority, Scope, and title of office	Deborah A. Snyder, Chief Information Security Officer
12/01/2020	Updated second paragraph of Scope to include all types of systems. Information Statement updated to current industry and NYS standards.	Karen Sorady, Chief Information Security Officer
05/20/2021	Updated Scope language	Karen Sorady, Acting Chief Information Security Officer

9.0 Related Documents

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

OWASP Proactive Controls

OWASP Mobile Security Project

OWASP Developer Cheat Sheets

OWASP Enterprise Security API

OWASP Secure Coding Practices – Quick Reference Guide

OWASP Secure Coding Practices Checklist

OWASP Source Code Analysis Tools

Common Weakness Enumeration (CWE) List

CWE/SANS Top 25 Most Dangerous Software Errors ‘CWE/SANS Top 25’)

Carnegie Mellon Software Engineering Institute CERT Secure Coding Standards

US Department of Homeland Security Build Security In

Exhibit 1: Coding Resources

Open Web Application Security Project) 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 organization 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 and developer cheat sheets. 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.

Additional information regarding OWASP, the ESAPI library and the Top Ten project is available at https://www.owasp.org/.

Common Weakness Enumeration

The CWE/SANS Top 25 Most Dangerous Software Errors publication is the result of collaboration between the SANS Institute, the MITRE Corporation, 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 Corporation 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 List.