UNIVERSITY
OF
CALIFORNIA

UCOP
ITS
Systemwide CISO Office
Systemwide IT Policy

UC Secure Software Development Standard

Revision History

Date:	By:	Contact Information:	Description:
4/02/18	Robert Smith	robert.smith@ucop.edu	Initial issue of the Standard. Approved by the CISOs for consideration by ITLC and shared governance. Interim until approved by ITLC.
7/31/19	Robert Smith	robert.smith@ucop.edu	Updated based on user feedback. This standard now cross-references the Secure Software Configuration Standard. In 4.2, “dynamic” was removed from the prohibition on SQL queries in server side code. Section 4.3 – Exception Handling was adjusted to add a case of the application appropriately handling the error. Reworded one requirement in 4.6 to make the Logging requirement clearer. Reworded one requirement for encryption in 4.7 to make the object clear.
8/21/19	Robert Smith	robert.smith@ucop.edu	Updated to conform to standard style sheet.
9/9/19	Robert Smith	robert.smith@ucop.edu	Updated 4.3 Catch block.
10/3/19	Robert Smith	robert.smith@ucop.edu	Approved by the ITLC.

Contents

1 Background and Purpose

This Standard defines the requirements for secure software development. These projects are sometimes called “custom,” “in-house” or “open-source” software applications.

This Standard must be used in conjunction with UC’s information security policy, BFB-IS-3 Electronic Information Security and UC’s Secure Software Configuration Standard.[1]

Full-featured and robust programming languages and development platforms can weaken our cyber defense if implemented incorrectly. The risk is so common that almost every penetration testing report contains a chapter on exploiting “application weaknesses.” Proper software development is required as part of the cyber risk management program.

Applications, regardless of where they are hosted or where they run, are often not secure by default and require specific steps to achieve a secure outcome. Many website samples and vendor materials focus on getting applications to work with a minimum investment of time and without accounting for ongoing support costs. All applications, regardless of how acquired, can represent significant cyber risks (e.g., free, open-source, cloud offerings, SaaS, downloaded, licensed, etc.). It is thus important to evaluate all applications for security risks.

Attackers can use poorly constructed applications to compromise UC Institutional Information and/or IT Resources or make it possible to use UC IT Resources to do harm to others.

As outlined in IS-3, security is part of the entire system lifecycle. The best time to start applying good security principles is before development when requirements are created as part of an overall security architecture.

There are many resources available to help select, design, architect, implement and test software and applications to ensure that they are secure. This Standard establishes a minimum set of practices to manage cyber risk. UISLs and IT Workforce Members should also consider reducing cyber risk through:

- Software and IT Resource architectures.
- Data minimization (e.g., collecting only the data needed, focusing on a specific purpose).
- Interface design and dependency management.

2 Scope

This Standard applies to all Locations. This Standard applies to all new software developed by or for the University of California as a network accessible production application:

- For business and administrative purposes;

[1] Other standards will often apply to developed software, customizations and extensions (e.g., Event Logging Standard, Account and Authentication Management Standard, Security Key and Certificate Management Standard, etc.).

- When contracts or grants set forth security requirements or concerns;
- When privacy or confidentiality is involved; and
- When there are regulatory requirements that include application security.

Note: This Standard does not apply to research computing; academic experiments; or student projects not otherwise covered above, provided there is no processing, storing or transmitting of Institutional Information classified at Protection Level 3 or higher.

Currently deployed or existing systems or applications that are processing, storing or transmitting Institutional Information classified at Protection Level 3 or higher must comply with this Standard when:

- A specific risk assessment indicates high risk to UC or data subjects.
- A major or substantive upgrade occurs.

3 Definitions and Key Terms

There are no specially defined terms required for using this Standard.

For more information about definitions, consult the IT Policy Glossary.

4 Requirements for Secure Software Development

All software benefits from its developer's adherence to secure software development practices regardless of the Protection Level (PL) or Availability Level (AL) classification associated with the Institutional Information processed, stored or transmitted.

Software does not exist in a vacuum (e.g., it is often colocated with other applications); a weakness in one application can become an attack vector to gain access to other applications or data. Secure practices for all software development are a vital part of adequate cyber risk management.

IT Workforce Members must follow secure software development practices during the entire software development lifecycle and implement controls appropriately. The layering of security controls helps prevent or detect breach attempts and can reduce the time required to detect and respond to attackers.

Even applications that process only Institutional Information classified at Protection Level 1 need to be secure so they are not exploited to attack users with malware downloads or redirects to malicious sites. Attackers can use vulnerabilities in these applications to gain access to internal IT Resources with a higher Protection Level.

4.1 Software Development Process

Secure software development includes integrating security in different phases of the software development lifecycle (SDLC), such as requirements, design, implementation and testing. The basic task of security requirement engineering is to identify and document actions needed for developing secure software systems. Security elements of the SDLC must include:

- Planning to meet security requirements and goals.
- Threat modeling.
- Design to include security and privacy concerns.

- System architecture (e.g., web, applications, user interfaces, programmatic interfaces, file import/export, reports, databases).
- Documentation.
- Change management (See IS-3, III).
- Testing, including creating test plans, reviewing test results and confirming fixes and patches.
- Secure deployment practices and separation of duties.

Regarding code review, Units developing software that will process, store or transmit Institutional Information classified at Protection Level 3 or higher or Availability Level 3 or higher must:

- Perform code reviews to reduce cyber risk.
- Consider and check for common security mistakes.
- Include in the review process a senior software developer and, if possible, choose an independent one.
- Include in the review process an IT Workforce Member with specific security experience.
- Use automated secure code testing/checking tools:
  - Perform static code analysis.
  - Perform dynamic code analysis.

4.2 Input Validation

IT Workforce Members developing software must:

- Validate user input before using the input data programmatically.
- Sanitize or reject invalid user input to protect against code injection attacks.
- Include user interface controls in the input validation strategy to make compliant and safe input easy for the user.
- Protect against buffer overflow attacks.
- Protect against array index errors.
- Protect against parameter manipulation attacks.
- Use parameterized SQL queries.
- Defend against SQL injection attacks.
- Put all SQL code/commands in server-side code.
  - Do not use concatenated database queries.
  - Do not put SQL in client-side code.
- Set Autocomplete=off in HTML to prevent the caching of sensitive information.
- Protect against URL query string manipulation attacks.

IT Workforce Members developing software must not use the following types of information in the URL/URI:

- Credentials.
- Access tokens, serial numbers or record numbers.
- PII (e.g., names, SSNs, driver's license numbers or dates-of-birth).
- PHI (e.g., a Medical Record Number, diagnosis, condition or name).

4.3 Exception and Error Handling

IT Workforce Members developing software must:

- Configure runtime environments so that they do not reveal native framework errors (e.g., .Net, J2EE) to the screen/browser.
- Set up custom error pages for framework errors.
- Handle expected errors and provide users/administrative staff enough detail in error messages to troubleshoot problems.
- Write code so there are no unhandled errors.
- Create (wrap code) last-resort error try-catch blocks.
- Never use empty catch blocks.
- Ensure catch blocks cause the application to do at least one of the following:
    - Stop running and exit in a secure state.
    - Appropriately handle the exception.
- Log detailed exception and error messages to the application event logs.

4.4 Cross Site Scripting (XSS) and Invalidated Redirects/Forwards

IT Workforce Members developing software must test untrusted data or code before sending it to a web browser. Testing must include proper validation for common XSS attacks.

Note: See the OWASP XSS (Cross Site Scripting) Prevention Cheat Sheet.
https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

4.5 Insecure Direct Object References

IT Workforce Members developing software must ensure that direct object references to any object, file, directory or database key include an access control check or other protection.

4.6 Logging

IT Workforce Members developing software must:

- Log events related to authentication (success or failure), use of privileged functions, administration activities and system/component starts and stops.
- Log security-related events to a separate log file when possible. This provides developers with additional flexibility for any specific rules related to the log file (e.g., special roles/permissions, chain-of-custody, etc.).
- Limit information in the log details, especially if security events are comingled with other application events (e.g., do not log sensitive information).
- Use file naming conventions (e.g., rotation, location, etc.) so the Location, Unit SIEM service or InfoSec professional can ingest the logs.
- Store security related logs separately from the application when possible.
- For Institutional Information and/or IT Resources classified at Protection Level 3 or higher or Availability Level 3 or higher, use a CISO-approved log monitoring service.
- Ensure sufficient storage is available to maintain logs throughout the required retention period.

4.7 TLS and Secure APIs

IT Workforce Members developing software must:

- Force HTTPS for all browser connections.
- Disable HTTP.

- Use TLS 1.2 or later to protect machine-to-machine connections (e.g., app server to DB server connections).
- Use strict transport security header.
- Encrypt Institutional Information classified at Protection Level 3 or higher using a separate and unique credential or key.
- Ensure that the encryption technology used is approved by the CISO.
- Authenticate and encrypt the session when software involves the transmission of Institutional Information classified at Protection Level 3 or higher.
- Encrypt APIs, restful interfaces, extract tools and service bus sessions.
- Authenticate APIs, restful interfaces, extract tools and service bus sessions.
- Separate public and private application APIs and configure CORS (Cross Origin Resource Sharing) headers to restrict invocation rights.

Note: IT Workforce Members must <u>not</u> write their own encryption code because it is too easy to make critical errors. Training does not exempt IT Workforce Members from this warning. This applies to everyone: DO NOT WRITE YOUR OWN CRYPTO. The problem of encrypting data has already been solved and vetted by the security community (e.g., FIPS 140-2 validated ciphers). See references below.

4.8 Credentials/Passphrases

IT Workforce Members developing software must:

- Never store user passphrases.
- Protect service account credentials with established tools and techniques.
- Never hardcode credentials.
- Implement lockout after 5 failed authentication attempts.
- Use secure protocols for credential or other secret exchanges (e.g., TLS 1.2 or later).

4.9 Session and Logout

IT Workforce Members developing software must:

- Ensure that session timeout is implemented.
- Make sure session tokens are random and long (GUID long).
- Change (delete old and create new) tokens at each logon and each privilege escalation.
- Delete, remove and invalidate tokens on logout.
- Use TLS 1.2 or later for all session tokens and cookies.
- Provide a logout function.
- Prominently display the logout function throughout the application.
- Properly close records, delete temporary objects and close operations on the server:
  - As part of the logout process.
  - After a set period of inactivity (e.g., less than one hour).
  - After a browser window close.

4.10 Federated Authentication / SAML / Shibboleth

IT Workforce Members developing software must use the Location-approved, single sign-on, SAML-based authentication or the CISO-approved method for authenticating users.

They must also:

- Assign permissions to user groups (not individual users).
- Assign users to user groups.
- Ensure the application has a user group with no/minimal privileges assigned.
- Assign or default new users to the user group with no/minimal privileges and make assignment to a group explicit.
- Check/recheck authorization at the next logical operation (e.g., for web applications at the next page request).
- Log all actions that grant users or groups permissions or rights.

4.11 File Management

IT Workforce Members developing software must:

- Prevent/disable directory traversal/directory listing.
- Filter web requests to block access to files with non-approved extensions (e.g., .dll, .config, .java, .cs, etc.).
- For IT Resources with Availability Level 3 or higher, set a directory quota or similar control.
- Ensure shared volumes/file shares have appropriate access restrictions limited to the application service account.

4.12 Secure Configuration

IT Workforce Members developing software must:

- Use secure configuration options for supporting technology, libraries, packages and tools.
- Perform or have performed a vulnerability scan of the entire solution and appropriately remediate findings based on risk before placing the solution into production.
- Secure the components used in an application.
- Run components with the least privilege possible.
- Keep components patched and up-to-date.
- Execute SQL with the least privilege possible.
- Use vetted and tested hardening guides to ensure the correct application of options and settings.
- Define, implement and maintain secure settings (e.g., do not rely on default settings).
- Secure the software configuration used in an application.
- Set the configuration to define which HTTPS methods (e.g., Get or Post) the application will support and whether HTTPS methods will be handled differently in different pages of the application.
- Set Header and Content-Security-Policy when possible.

4.13 Documentation

IT Workforce Members developing software with Protection Level 3 or higher or Availability Level 3 or higher must create project documentation to:

- Record the security features and steps taken to protect:
  - Confidentiality concerns.
  - Integrity concerns.
  - Availability concerns.
- Track defects and fixes to defects.
- Note/record testing processes and how testing processes manage cyber risk.
- List components used and the security state of the components.

4.14 Version Control

IT Workforce Members developing software with Protection Level 3 or higher or Availability Level 3 or higher must:

- Use a software version control system or repository.
- Deploy test and production applications from the version control system or repository.
- Tag/label versions so that they can be extracted at a later time.
- Configure the version control system or repository to prevent and detect unauthorized changes.

5 References

UC Policy

Business and Finance Bulletin IS-3 – Electronic Information Security

Business and Finance Bulletin IS-12 – Continuity Planning and Disaster Recovery

UC Standards

IT Security Committee – UC Account and Authentication Management Standard

IT Security Committee – UC Event Logging Standard

IT Security Committee – UC Minimum Security Standard

IT Security Committee – UC Secure Software Configuration Standard

External Resources

Howard, M. and LeBlanc, D. (2004). *Writing Secure Code 2, Practical Strategies and Proven Techniques for Building Secure Applications in a Networked World (Developer Best Practices)*. Redmond, Washington: Microsoft Press.

OWASP Top 10:
https://www.owasp.org/index.php/Top_10_2017-Top_10

OWASP Secure Coding Guidelines:
https://www.owasp.org/images/0/08/OWASP_SCP_Quick_Reference_Guide_v2.pdf

OWASP Web.config Encryption:
https://www.owasp.org/index.php/OWASP_Backend_Security_Project_.NET_Security_Programming#Web.config_Encryption

OWASP XSS (Cross Site Scripting) Prevention Cheat Sheet:
https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

FIPS 140-2: https://csrc.nist.gov/Projects/Cryptographic-Module-Validation-Program/Standards

Cryptographic Algorithm Validation Program: https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/validation

NIST Special Publication 800-111, Guide to Storage Encryption Technologies for End User Devices: https://csrc.nist.gov/publications/detail/sp/800-111/final

Developing for accessibility: http://www.ucop.edu/electronic-accessibility/standards-and-best-practices/description-of-wcag-2.0.html