Baldwin Wallace University Information Technology Standard

Issued by:	**Information Technology**
Title:	**Secure Software Development**
Number:	**ITS-BW-27-01**
Publish date:	**September 1, 2024**

NIST SSDF:

Due to BW's very limited software development and small staff, many of the in-depth recommendations by NIST and OWASP are simply not practical. What follows are BW's requirements to follow each of the NIST SSDF best practices.

ITS-BW-27-01 Secure Software Developmer

NIST SSDF Best Practices	BW Actions to be Taken
Define Security Requirements for Software Development (PO.1): Ensure that security requirements for software development are known at all times so that they can be taken into account throughout the SDLC and duplication of effort can be minimized because the requirements information can be collected once and shared. This includes requirements from internal sources (e.g., the organization's policies, business objectives, and risk management strategy) and external sources (e.g., applicable laws and regulations).	To ensure security requirements are well understood the data affected must be classified and protected per the DGP-BW-04 Data Classification Policy. Additionally, consideration must be taken into account based on the biannual organization's IT risk assessment. For purchased applications or code developed for the university by a third party, Baldwin Wallace University leverages contractual agreements and the industry-standard "Higher Education Community Vendor Assessment Toolkit" (HECVAT) for vendor risk assessments. The HECVAT is a questionnaire framework specifically designed for higher education to measure vendor risk. See ITP-BW-16 Outsourcing Policy and ITS-BW-16-02 Vendor Risk Assessment for further information.
Implement Roles and Responsibilities (PO.2): Ensure that everyone inside and outside of the organization involved in the SDLC is prepared to perform their SDLC-related roles and responsibilities throughout the SDLC.	All code developed by BW IT may only be moved into production after it has been approved by management using BW's change control process as defined in the ITP-BW-18 Change Management Policy. Code development responsibilities for BW IT must be defined in relevant job descriptions. Due to the limited size of the BW IT staff, typically only one person is familiar with a product or system making it not operationally feasible to perform segregation of duties. As such, it has been deemed an acceptable risk for one person to perform all SDLC tasks.

Implement Supporting Toolchains (PO.3): Use automation to reduce human effort and improve the accuracy, reproducibility, usability, and comprehensiveness of security practices throughout the SDLC, as well as provide a way to document and demonstrate the use of these practices. Toolchains and tools may be used at different levels of the organization, such as organization-wide or project-specific, and may address a particular part of the SDLC, like a build pipeline.	BW currently does not leverage third-party tools supporting code development as most all code development is based on a specific platform's requirements such as Colleague which uses a custom development environment. As such this requirement is nonapplicable. Should this situation change, this standard will be updated.
Define and Use Criteria for Software Security Checks (PO.4): Help ensure that the software resulting from the SDLC meets the organization's expectations by defining and using criteria for checking the software's security during development.	Record security check approvals, rejections, and exception requests as part of the workflow and tracking system. Only allow authorized personnel to access the gathered information, and prevent any alteration or deletion of the information.
Implement and Maintain Secure Environments for Software Development (PO.5): Ensure that all components of the environments for software development are strongly protected from internal and external threats to prevent compromises of the environments or the software being developed or maintained within them. Examples of environments for software development include development, build, test, and distribution environments.	All code development must take place in test or development environments. Never in production unless specifically approved by management via Change Control. All test or development environments must be secured, hardened, and monitored to the same level as their production equivalent environments. Any exception must be approved by the CIO. Access to test or development environments must be managed by the principle of least privilege.
Protect All Forms of Code from Unauthorized Access and Tampering (PS.1): Help prevent unauthorized changes to code, both inadvertent and intentional, which could circumvent or negate the intended security characteristics of the software. For code that is not intended to be publicly accessible, this helps prevent theft of the software and may make it more difficult or time-consuming for attackers to find vulnerabilities in the software.	Store all forms of code – including source code, executable code, and configuration-as-code – based on the principle of least privilege so that only authorized personnel, tools, services, etc. have access. Where technically and financially reasonable: - Require MFA. - Leverage BW's Privileged Access Management solution. - Use BW-approved code signing certificates.
Provide a Mechanism for Verifying Software Release Integrity (PS.2): Help software acquirers ensure that the software they acquire is legitimate and has not been tampered with.	Not applicable. BW does not sell or share its code or resulting products.

Archive and Protect Each Software Release (PS.3): Preserve software releases in order to help identify, analyze, and eliminate vulnerabilities discovered in the software after release.	Securely archive the necessary files and supporting data (e.g., integrity verification information, provenance data) to be retained for each software release. Where possible, use a code version library solution.
Design Software to Meet Security Requirements and Mitigate Security Risks (PW.1): Identify and evaluate the security requirements for the software; determine what security risks the software is likely to face during operation and how the software's design and architecture should mitigate those risks; and justify any cases where risk-based analysis indicates that security requirements should be relaxed or waived. Addressing security requirements and risks during software design (secure by design) is key for improving software security and also helps improve development efficiency.	Developers will: - Be trained in FERPA, FTC Safeguards Rules, and other relevant requirements. - Understand the data classification of all data involved in the coding project before generating any coding. - Understand the security capabilities and limitations of the platform being customized. - Maintain applicable records of design decisions, risk responses, and approved exceptions that can be used for auditing and maintenance purposes throughout the rest of the software life cycle.
Review the Software Design to Verify Compliance with Security Requirements and Risk Information (PW.2): Help ensure that the software will meet the security requirements and satisfactorily address the identified risk information.	Due to the limited size of the BW IT staff, typically only one person is familiar with a product or system making it not operationally feasible to perform compliance reviews. As such, it has been deemed an acceptable risk to not perform manual code reviews. If a software code review tool is available for the type of code being generated, then it will be used if technically possible.
Reuse Existing, Well-Secured Software When Feasible Instead of Duplicating Functionality (PW.4): Lower the costs of software development, expedite software development, and decrease the likelihood of introducing additional security vulnerabilities into the software by reusing software modules and services that have already had their security posture checked. This is particularly important for software that implements security functionality, such as cryptographic modules and protocols.	Using source code from the Internet must only be downloaded from original trusted sites. The downloading of code from mirror sites or other sites is prohibited. Any code that is downloaded must be manually reviewed by the developer to ensure it is well-understood and free from any malicious code. Any code that is downloaded must be kept in a library for future reference while that code is in use.

Create Source Code by Adhering to Secure Coding Practices (PW.5): Decrease the number of security vulnerabilities in the software, and reduce costs by minimizing vulnerabilities introduced during source code creation that meet or exceed organization-defined vulnerability severity criteria.	Developers must follow all secure coding practices that are appropriate to the development languages and environment to meet the organization’s requirements. Examples include, but are not limited to: - Validate all inputs, and validate and properly encode all outputs. - Avoid using unsafe functions and calls. - Detect errors, and handle them gracefully. - Provide logging and tracing capabilities. - Follow procedures for manually ensuring compliance with secure coding practices when automated methods are insufficient or unavailable. - Use tools (e.g., linters, formatters) to standardize the style and formatting of the source code. - Check for other vulnerabilities that are common to the development languages and environment. - Review their human-readable code.
Configure the Compilation, Interpreter, and Build Processes to Improve Executable Security (PW.6): Decrease the number of security vulnerabilities in the software and reduce costs by eliminating vulnerabilities before testing occurs.	Use up-to-date versions of compiler, interpreter, and build tools. Follow change management processes as defined in ITP-BW-18 Change Management Policy when deploying or updating compiler, interpreter, and build tools, and audit all unexpected changes to tools. Perform all builds in a test or developed environment. Never production unless specifically approved by management via Change Control.
Review and/or Analyze Human-Readable Code to Identify Vulnerabilities and Verify Compliance with Security Requirements (PW.7): Help identify vulnerabilities so that they can be corrected before the software is released to prevent exploitation. Using automated methods lowers the effort and resources needed to detect vulnerabilities. Human-readable code includes source code, scripts, and any other form of code that an organization deems human-readable.	Due to the limited size of the BW IT staff, typically only one person is familiar with a product or system making it not operationally feasible to perform compliance reviews. As such, it has been deemed an acceptable risk to not perform manual code reviews. If a software code review tool is available for the type of code being generated, then it will be used if technically possible.
Test Executable Code to Identify Vulnerabilities and Verify Compliance with Security Requirements (PW.8): Help identify vulnerabilities so that they can be corrected before the software is released in order to prevent exploitation. Using automated methods lowers the effort and resources needed to detect vulnerabilities and improves traceability and repeatability. Executable code includes binaries, directly executed bytecode and source code, and any other form of code that an organization deems executable.	All in-scope platforms must maintain at a minimum a test and production environment. All newly developed code must be fully tested and evaluated in the test environment where technically possible. Any exception must be approved by the CIO.

Configure Software to Have Secure Settings by Default (PW.9): Help improve the security of the software at the time of installation to reduce the likelihood of the software being deployed with weak security settings, putting it at greater risk of compromise.	Define a secure baseline for the platform being used for development by determining how to configure each setting that affects security or a security-related setting so that the default settings are secure and do not weaken the security functions provided by the platform, network infrastructure, or services. Conduct annual auditing to ensure that the settings, including the default settings, are working as expected and are not inadvertently causing any security weaknesses, operational issues, or other problems. Store the default configuration in a usable format and follow change control practices for modifying it (e.g., configuration-as-code).
Identify and Confirm Vulnerabilities on an Ongoing Basis (RV.1): Help ensure that vulnerabilities are identified more quickly so that they can be remediated more quickly in accordance with risk, reducing the window of opportunity for attackers.	In-scope platforms will be scanned per the ITP-BW-22 Patch and Vulnerability Management Policy and be included in penetration testing. Additionally, BW will subscribe to relevant vendor and publicly available information feeds of vulnerabilities. Any security incident will be handled by the BW Incident Response team per the ITP-BW-10 Incident Response Policy and its supporting incident response plans.
Assess, Prioritize, and Remediate Vulnerabilities (RV.2): Help ensure that vulnerabilities are remediated in accordance with risk to reduce the window of opportunity for attackers.	In scope, platforms will be evaluated and remediated per the ITP-BW-22 Patch and Vulnerability Management Policy.