# Brink’s Secure Systems/Software Development Lifecycle Standard

# Contents

# I. Background, Purpose and Scope

## A. Background

The Secure Software or System Development Lifecycle (SSDLC) is used to ensure that data security is adequately considered and built into each phase of every Software or System Development Lifecycle (SDLC).

Information systems are in a constant state of evolution with upgrades to hardware, software, firmware, and possible modifications in the surrounding environment. The SSDLC defines data security requirements and tasks that must be considered and addressed within every system, project, or application that is created or updated to address a business need. The SSDLC assures that our SDLCs encompasses the entire lifespan of a system, from conceptualizing through implementing, operating, and finally decommissioning.  This standard focuses on a general overview of security integration into the SDLC and does not prescribe or mandate any particular SDLC model or methodology.

## B. Purpose

The idea is to have security built-in rather than bolted on, maintaining the security paradigm during every phase, to ensure a secure SDLC. The purpose of this document is to provide organization-wide security practices for secure software and system development when creating and maintaining Brink’s developed and acquired assets and applications.

## C. Scope

The standard applies to all Brink’s personnel irrespective of status, including temporary staff, contractors, consultants, and third parties who have access to Brink’s data and systems.

There may be circumstances where it is not possible to fully apply the standard.  In such cases, a policy Exception Request must be raised, and approval sought from Information Security Management.

# II. Global Information Security Policy Statements

- Brink’s shall periodically perform, at least annually, dynamic application vulnerability assessments on both internal and external applications. Results of the application testing shall be provided to the development team in a summary report and include vulnerability detail and remediation plan.

- Brink’s follows OWASP Secure Coding Guidelines for Web Application Secure Coding Practices and utilizes Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) for both internal and external web applications, to ensure that processes for maintaining application security testing adhere to industry best practices.

- Assessments shall be completed prior to releasing any new applications or significant enhancements to existing applications to production.

- All personnel in a developer role must complete formal training in general secure coding techniques and in developing secure code in the programming language(s) they use. Annual OWASP training must be completed by developers to create awareness on the latest threat vectors in web applications. OWASP training must be completed within 45 days of hire.

- External and Internal security scans of all domains facing IT assets are performed by a third-party vendor or internal resource according to an agreed upon schedule (at least annually). Threats and vulnerabilities are documented, and remediation is executed in a timely manner to comply with internal and contractual agreements. All Critical and High severity items must be remediated prior to production release.

- At least quarterly, internal security scans of all public domain facing IT assets are performed internally according to an agreed upon schedule. Threats and vulnerabilities are documented and evaluated for remediation based on risk level and mitigated in a timely manner to comply with current requirements. All Critical and High severity items must be remediated prior to production release.

- External and internal vulnerability scans are performed outside of the network and target specific IP, Web application and all publicly facing Infrastructure, this is used to measure and test potential exposure over the internet.

- Brink’s has a process to rescan findings from vulnerability scans/penetration tests until no critical or high-risk findings are identified.

- Record the response to each risk, including how mitigations are to be achieved and what the rationales are for any approved exceptions to the security requirements. Add any mitigations to the software’s security requirements.

- Periodically re-evaluate all approved exceptions to the security requirements, and implement changes as needed.

- All development, acquisition, or integration projects for information system resources, whether performed in house or by a business partner, must follow an approved systems development life-cycle methodology inclusive of security activities. Security must be addressed throughout the information system resource life-cycle process, from requirements, design, build, system testing, acceptance testing, release (and production) and retirement. All systems development must follow secure coding best practices.

- Each information system resource (i.e., application or infrastructure component) must receive Brink’s certification and accreditation. The Brink’s certification and accreditation process defines a formal review process that ensures adequate security is incorporated during each phase of the project life- cycle.

# III. Standard Statements

## General

All managers must ensure that this standard is implemented and complied within their functional areas. All personnel must adhere to this standard regardless of their status. Information Security Management must maintain this Standard and provide advice on implementation.  The standard must be reviewed annually.

## Phase 1: Requirements & Analysis

The software development process starts with gathering requirements and systems analysis, the results of which are then used to create the design.

- Ensure that security and compliance requirements for software development are always known and documented; so that they can be integrated considered throughout the Secure Software Development Lifecycle (e.g., the organization’s policies, business objectives, and risk management strategy) and external sources (e.g., applicable laws and regulations).
- Use data classification methods to identify and characterize each type of data that the software will interact with.
- The compliance requirements should be mapped to the security controls.
- Determine if encryption is required based on data classification.
- GIS Review/Sign-off

## Phase 2: Design

Software design is the blueprint of the system which includes taking all the security requirements into consideration. The components in the design are structured by utilizing software modules/functions/libraries. Architects follow recommended security practices to deploy, operate, and maintain software.

- Threat modeling is performed, which puts the software through various scenarios of misuse to assess the security robustness. In the process, various avenues to tackle potential problems emerge.
- Software design includes security and privacy controls previously identified in the requirements phase.
- Architects utilize industry best practice guidelines, such as OWASP to embed security controls in the architecture design. (e.g., web, applications, user interfaces, programmatic interfaces, file import/export, reports, databases).
- Maintain 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.

## Phase 3: Development/Coding

It is necessary to establish secure coding practices among developers through guidelines and awareness campaigns. The best practices in the coding phase of a secure SDLC revolve around educating the developers. Development best practices provides developers an insight into how security vulnerabilities are created. These include not just technical vulnerabilities, but also problems from a business logic perspective. A source code review helps in making sure the coding quality is maintained, in addition to meeting secure coding standards.

- Train the development team (security champions, in particular) or collaborate with a risk modeling expert to create models and analyze how to use a risk-based approach to communicate the risks and determine how to address them, including implementing mitigations.
- Perform the code review and/or code analysis based on the organization's secure coding standards, and record and triage all discovered issues and recommended remediation in the development team's workflow or issue tracking system.
- Perform more rigorous assessments (Static Application Security Testing) for high-risk areas, such as protecting sensitive data and safeguarding identification, authentication, and access control, including credential management.
- Record and triage all discovered issues, vulnerabilities, and recommended remediation activities in the development team's workflow or issue tracking system.
- Fix/Remediate security findings.
- For web applications, as applicable, use an established certificate authority for code signing so that customers' operating systems or other tools and services can confirm the validity of signatures before use.
- Data should be masked or sanitized before being implemented into a working environment.

- GIS Review/Sign-off of SAST results

## Phase 4: System Testing/Quality Assurance (QA)

System testing is an assessment of the software in terms of threats, risks, and vulnerabilities. Testers examine the software to see whether the system follows compliance according to the given requirements. They also examine for flaws that expose sensitive and confidential data to hackers or make the software susceptible to vulnerabilities. System testing may include phases such as unit testing and integration testing.

Quality assurance testing reviews, analyzes, and/or tests the software's code to identify or confirm the presence of previously undetected vulnerabilities. If testers find any such flaws, the developers can secure them with coding. The three pillars of quality are security, operability, and features.

- Scope the testing, design the tests, perform the testing, and document the results, including recording and triaging all discovered issues and recommended remediation in the development team's workflow or issue tracking system.
- Define a secure baseline by determining how to configure each security-related setting. Therefore, the security defaults settings do not weaken the security functions provided by the platform, network infrastructure, or services.
- Perform Dynamic Analysis Security Testing (DAST) to help find potentially exploitable vulnerabilities while running in production.
- Analyze each vulnerability to gather sufficient information about risk to plan its remediation or other risk response.
- Record the security vulnerabilities (if any) to serve as issues that need to be remediated according to risk severity (e.g., in the software specification, in the issue tracking system, in the threat model).
- Prior to production, at least every 12 months, and after significant changes, use penetration testing to simulate how an attacker might attempt to compromise the software in high-risk scenarios.
- GIS Review/Sign-off of DAST results
- GIS Review/Sign-off of PenTest results as applicable

## Phase 5: User Acceptance Testing (UAT)

User Acceptance Testing arises once software has undergone Unit, Integration, and System. UAT is a testing methodology where the end-user is included in testing, to approve the system/product according to their specifications before passing it to the production environment. UAT test plans document entry and exit criteria for user acceptance testing, test scenarios, and test cases approach, and timelines of testing.

UAT is the final phase to prevent exploitation, which includes verification of embedded security, questioning the quality of the software, and prioritizing security by using tools to measure software vulnerabilities.

- Review the software design to verify compliance with security requirements are clearly understood.

- Ensure that the software will meet the security requirements and adequately address the identified risk information.
- Create a user acceptance test plan that outlines the strategy that will be used to verify and ensure an application meets its business and security requirements.
- Data should be scrambled/obfuscated for privacy and security reasons.
- Run and record the results by executing test cases, record the security vulnerabilities (if any), and report bugs (if any).
- Re-test once fixed/remediated. Test Management tools can be used for execution.
- Deliverables for user acceptance testing are test plans, user acceptance testing scenarios and test cases, test results, and vulnerability/defect logs.
- Confirm with the business unit that the system requirements are met and require a sign-off after the user acceptance testing is completed.
- GIS Review/Sign-off of UAT Test Plan for security requirements.

## Phase 6: Deployment / Maintain

In the final deployment phase of a Secure Software Development Lifecycle, the different components of the system/software interact with each other. Documenting information system changes and assessing the potential impact of these changes on the security of a system are essential activities to assure continuous monitoring and prevent lapses in the system security accreditation.

- Continuously monitor the performance of the system to ensure that it is consistent with pre-established user and security requirements.
- Control and Configuration management of activities should be conducted to document any proposed or actual changes in the security plan of the system.
- Review the SDLC standard and update it if appropriate to prevent (or reduce the likelihood of) the root cause recurring in updates to the software or in new software that is created.

## Phase 7: Purchase/Acquired Software Hardening

Third-party software often comes with many security risks; therefore, one of the best practices for system hardening involves updating them regularly to ensure not remaining accessible to vulnerabilities.

Updates, patches, and enhancements to the third-party application code are constantly required. It is a cycle that repeats itself, but security, even at the time of these modifications, must always be in focus to ensure a robust and secure SDLC.

- Perform party security risk management assessment (work with compliance).

- Gather information from software acquirers, users, and public sources on potential vulnerabilities in the software and third-party components that the software uses and investigate all credible reports.
- Review and evaluate third-party software components in the context of their expected use. Designate which security components must be included in software.
- If possible, perform application scanning/testing on the executable code to identify vulnerabilities and verify compliance with security requirements.
- Record the response to each risk, including how mitigations are to be achieved and what the rationales are for any approved exceptions to the security requirements. Add any mitigations to the software’s security requirements.
- GIS Review/Approval to purchase/implement third-party software.

# IV. Definitions

- Dynamic Application Security Testing (DAST) is the process of analyzing a web application through the front-end to find vulnerabilities through simulated attacks.
- Static Application Security Testing (SAST) identifies the root cause of vulnerabilities and helps remediate the underlying security flaws.
- Architecture Review Board (ARB) serves as a governance body, defines appropriate IT strategies, technical design standards, and ensures system development alignment with those strategies.
- Penetration Testing (PenTest) is an authorized simulated attack performed on a computer system to evaluate its security. Penetration testers use the same tools, techniques, and processes as attackers to find and demonstrate the business impacts of weaknesses in a system.
- Web application Scan refers to a set of services used to detect various security issues with web applications and identify vulnerabilities and risks, including SQL injection, cross-scripting, web-server configuration, etc.

# V. Compliance

Compliance with this document is mandatory for the Executive Department including all executive offices, and all boards, commissions, agencies, departments, divisions, councils, and bureaus. Violations are subject to disciplinary action in accordance to applicable employment and collective bargaining agreements, up to and including the termination of their employment and/or assignment with the Brink’s. Exceptions to any part of this document must be requested via email to the GRC Team. A policy exception may be granted only if the benefits of the exception outweigh the increased risks, as determined by the Brink’s CISO.

# Appendix 1: Document Control

| | |
|---|---|
| Title | Brink’s Secure Systems/Software Development Lifecycle Standard |
| Version | 1.0 |
| Date Issued | 2/20/2023 |
| Status | Final |
| Document Owner | Carrie Rogers, Global IT Governance Manager |
| Authorized By | Patrick Benoit, Global CISO |
| Creator Organization Name | Global Information Security |
| Document Category | Global Information Security Standard |
| Document Classification | Confidential |

## Document Revision History

| Version | Date | Author | Reviewer | Summary of changes |
|---|---|---|---|---|
| 1.0 | 2/20/2023 | Carrie Rogers | Patrick Benoit | Initial release |
| | | | | |
| | | | | |
| | | | | |
| | | | | |