System Acquisition and Development Lifecycle Policy

Version 1 - Approved by Rishabh

Contents

1. Objective
2. Scope
3. Policy Statement
4. System Acquisition, Development & Maintenance
5. Security in Development & Support Processes
6. System Testing
7. Test Data
8. Document Security Classification
9. Non-Compliance
10. Responsibilities
11. Schedule
12. Version history

1. Objective

The objective of this policy is to provide a framework to ensure that software development activities performed in Superhawk AI are aligned with integrated information security considerations throughout the development lifecycle. Superhawk AI is committed to developing robust systems which are reliable while ensuring that security is an integral part of information systems throughout all phases of acquisition, development, and maintenance.

2. Scope

This policy applies to all software development and related activities undertaken by Superhawk AI, including services delivered as SaaS. It covers work performed by Superhawk AI employees, contractors, and third-party vendors that interact with source code, build pipelines, testing environments, or production deployments. The scope includes code stored in GitHub, builds and deployments on AWS and GCP, and data stored in cloud platforms or on employee devices (laptops/workstations).

3. Policy Statement

Superhawk AI shall ensure that security is integral to its information systems throughout all phases of the acquisition, development, and maintenance life cycle. Security should be considered at every stage of an information system's life cycle (e.g., feasibility, planning, development, implementation, maintenance, retirement, and disposal) to:

- Ensure conformance with all appropriate security requirements.
- Protect enterprise data throughout its life cycle of system development.
- Prevent the introduction of new risks when the system is modified.
- Ensure proper removal/disposal of data when the system is retired.

4. System Acquisition, Development & Maintenance

4.1 System Development Life Cycle

Security shall be considered and included in all phases of the Software development lifecycle, including Requirement analysis, Design, Development, Testing, Implementation, Operations, and Maintenance.

4.2 Requirement Gathering

- Security and privacy requirements needed for the new product or application shall be defined at the requirements definition stage.
- Legal and regulatory implications and security requirements related to confidential data collection and usage done by the proposed system shall be considered.
- Superhawk AI shall consider requirements for ensuring the reliability and availability of information systems. Where availability cannot be guaranteed using existing architecture, redundant components or alternative architectures should be considered.

4.3 System Design

- It is important to identify threats and potential vulnerabilities early in the design phase of the software lifecycle. Areas of system misuse and ways in which protective measures could be bypassed shall be identified.
- The operating environment, internal and external interfaces of the system, sub-systems and components, data input and output from these sub-systems, and how the components of the software work together should be identified.
- Software shall be designed to operate with minimum privileges necessary.
- Application permissions, privileges, and access controls shall be designed to strictly adhere to the user roles defined.

4.4 System Development

- Superhawk AI uses GitHub for source code management and AWS and GCP for build and deployment infrastructure; the development environment must be logically separated from production using branch protections, environment-specific credentials, and isolated cloud projects/accounts.
- During system development, developers shall be instructed to observe caution in the below areas:
    - Check the validity of incoming data.
    - Check the validity of outgoing data.
    - Adhere to memory management best practices.
    - Secure practices during authentication and session management.
    - Use best practices for errors and exception management.
- Source code shall be stored in GitHub with role-based access controls and branch protection rules. Code versioning and deployments shall be governed by defined CI/CD pipelines; secrets must not be stored in repositories and shall be managed using approved secret management solutions or cloud provider secret stores.

5. Security in Development & Support Processes

5.1 Secure Application Development Principles

As a part of secure development principles, Superhawk AI shall consider:

- Best practices and latest libraries for each programming language used.
- Security in the application version control and code repository.
- Training developers on the secure coding aspects.
- Ensure developers' capability of avoiding, finding, and fixing vulnerabilities wherever possible.

5.2 Security Requirements for Information Systems

- Changes to systems within the development lifecycle should be controlled by the use of formal change control procedures.
- The introduction of new systems and major changes to existing systems should follow a formal process of documentation, specification, testing, managed implementation, and quality control.
- This process should include an analysis of the impacts of changes and the specification of security controls needed.
- This process should also ensure that existing security procedures are not compromised, that support programmers are given access only to those parts of the system necessary for their work, and that formal agreement and approval for any change is obtained.
- During change control procedures the following diligence is to be considered:
    - Ensuring changes are submitted by authorized users.
    - Reviewing controls and integrity procedures to ensure that they shall not be compromised by the changes.
    - Identifying all software, information, database entities, and hardware that requires amendment.
    - Identifying and checking critical code to minimize the likelihood of known security weaknesses.
    - Ensuring authorized users approve changes prior to implementation.
    - Ensuring that the system documentation is updated on the completion of each change, wherever applicable.
    - Maintaining version control for all software updates.
    - Maintaining a record of all change requests.
    - Ensuring that Standard Operating Procedures and user manuals are changed as necessary to remain appropriate, as applicable.
    - Ensuring that the implementation of changes takes place at the right time and does not disturb the business processes involved.
    - Testing of new software should be done in an environment segregated from both the production and development environments. The tests should include patches, service packs, and other updates.
    - Automated updates should not be used on critical systems as some updates can cause critical information systems to fail.
    - Where automatic updates are considered, the risk to the integrity and availability of the system should be weighed against the benefit of speedy deployment of updates.
    - Technical review of applications after operating platform changes.
    - When underlying operating platforms are changed, business-critical applications should be reviewed and tested to ensure there is no adverse impact on organizational operations or security.
- As far as possible and practicable, vendor-supplied software packages should be used without modification. Where a software package needs to be modified, the following points should be considered:
    - The risk of built-in controls and integrity processes being compromised.
    - Whether the consent of the vendor should be obtained.
    - The possibility of obtaining the required changes from the vendor as standard program updates.
    - The impact if the organization becomes responsible for the future maintenance of the software as a result of changes.
    - Compatibility with other software in use. If changes are necessary, the original software should be retained, and the changes applied to a designated copy.
    - A software update management process should be implemented to ensure the most up-to-date approved patches and application updates are installed for all authorized software.
    - All changes should be fully tested, so that they can be reapplied, if necessary, to future software upgrades. If required, the modifications should be tested and validated by an independent evaluation body.

5.3 Secure Engineering

- Security system engineering principles include security at all the architecture layers (business, data, applications, and technology), balancing the need for information security with the need for accessibility.
- New technology should be analyzed for security risks and the design should be reviewed against known attack patterns.
- Systems should be regularly reviewed to ensure that they remain up to date to address any new potential threats and be scalable.
- Security engineering principles should be applied, where applicable, to outsourced information systems through the contracts and other binding agreements between the organization and the third party to whom the organization outsources.

5.4 Establishing Secure Development Environments

- The Engineering team shall ensure development and testing environments are provisioned in separate AWS/GCP accounts or projects, with least-privilege access controls, network segmentation, and logging enabled. Access to these environments shall be reviewed periodically.
- A secure development environment includes people, processes, and technology associated with system development and integration.
- Engineering shall rely on cloud provider-managed backups and encryption for cloud resources; where backups or encryption are vendor-managed, the organization will maintain documented evidence of vendor controls and configurations.

- Engineering team should assess risks associated with individual Information system development efforts and provide requirements for secure development environments for specific system development efforts, considering:
  - Sensitivity of data to be processed, stored, and transmitted by the system.
  - Applicable external and internal requirements, e.g., regulations or policies.
  - Security controls already implemented by the Superhawk AI that supports information system development.
  - Trustworthiness of personnel working in the environment.
  - Degree of outsourcing associated with system development.
  - The need for segregation between different development environments.
  - Control of access to the development environment.
  - Backups should be stored at secure offsite locations.
  - Control over the movement of data from and to the environment.

6. System Testing

- New and updated systems require thorough testing and verification during the development processes, including the preparation of a detailed schedule of activities and test inputs and expected outputs under a range of conditions.
- For in-house developments, initial tests shall be performed by the Engineering team. Independent acceptance testing shall be performed by a designated QA function or an independent reviewer before deployment to production. Automated unit, integration, and security tests (including static analysis and dependency scanning) should be integrated into CI/CD pipelines hosted in GitHub and run against isolated test environments in AWS/GCP.
- The extent of testing should be decided by the Engineering team in concurrence with business requirements considering the importance and nature of the system.
- The Engineering team shall use approved automated code analysis and vulnerability scanning tools; identified defects must be tracked in the organization's issue tracker and remediated according to severity-based SLAs.
- Testing should be performed in a test environment to ensure that the Information system shall not introduce vulnerabilities to the Superhawk AI environment and that the tests are reliable.

7. Test Data

- The use of operational data containing personally identifiable information or any other confidential information for testing purposes should be avoided.
- If personally identifiable information or otherwise confidential information is used for testing purposes, all sensitive details and content should be protected by removal or modification.
- The following guidelines should be applied to protect operational data when used for testing purposes:
  - The access control procedures, which apply to production application systems, should also apply to test application systems.
  - Operational information should be erased from a test environment immediately after testing.
  - The copying and use of operational information should be logged to provide an audit trail.
- Where possible, anonymized or synthetic data shall be used for testing. Any use of production data for testing requires documented approval from the Data Protection Officer or designated owner and must follow masking or anonymization procedures prior to use.

8. Document Security Classification

Company Internal (please refer to the Data Classification policy for more details).

9. Non-Compliance

Compliance with this policy shall be verified through various methods, including, but not limited to, automated reporting, audits, and feedback to the policy owner. Any staff member found to be in violation of this policy may be subject to disciplinary action, which may include termination of employment or contractual agreement. The disciplinary action shall depend on the extent, intent, and repercussions of the specific violation.

10. Responsibilities

The Information Security Officer is responsible for approving and reviewing policy and related procedures. Supporting functions, departments, and staff members shall be responsible for implementing the relevant sections of the policy in their area of operation.

11. Schedule

This document shall be reviewed annually and whenever significant changes occur in the organization, such as changes to infrastructure (AWS/GCP), source code management (GitHub), major regulatory scope (SOC 2, ISO 27001, GDPR), or the introduction of new third-party vendors.

End of System Acquisition and Development Lifecycle Policy. For version history, please see the next page.

Version History

	Version	Log	Date
1	Current	Policy version approved by Rishabh	13 Mar, 2026
1		New policy version created	13 Mar, 2026