Operations Security Procedure

Version 1 - Approved by Youssef Ouyhya

Contents

1. Objective
2. Scope
3. Operational Security Procedure
4. Document Security Classification
5. Non-Compliance
6. Responsibilities
7. Schedule
8. Version history

1. Objective

The objective of this procedure is to provide guidelines to ensure the operational security of CNTXT's services through procedures for backup, change management, logging, and vulnerability management.

2. Scope

This procedure covers all systems within our production environment. The production environment includes all cloud assets used in hosting and its subdomains.

3. Operations Security Procedure

3.1. Change Management Procedure

3.1.1. Use of Version Control Systems

- All software developed in the service of CNTXT or any subdomain of CNTXT's products should be version controlled i.e. the latest version of our software as well as any previous version of our software are readily available.
- CNTXT uses a decentralized version control system like git for code changes. This allows engineers to work on bug fixes, new feature development, and other independent projects simultaneously. Before synchronizing with the central repository, it is recommended that engineers work on local branches created from an appropriate version of the central repository. All changes must be tested locally before the changes are deployed to users.

3.1.2. Initiating Planned Changes

- While developing new features in CNTXT's products, it is recommended to start a new feature branch in git. All requirements and specifications of a feature may not be known at the beginning of the development of the feature. One can create new branches of the feature branch to develop sub-features as necessary.
- Most feature branches exist on the local systems of developers working on the feature. They need not be synced with the central, company-wide, repository. However, when a feature is considered ready to be used by customers, a pull request is created. Any developer or software engineer can initiate a pull request.
- Alternatively, for other changes like network changes, it is recommended to be tracked in a ticketing platform where change its owner, approver, its impact and details steps for rollback are well documented

3.1.3. Approving Planned Changes

- A pull request outlines the differences in code that this feature proposes. A pull request has to be reviewed by other peer developers or managers. All pull requests should be reviewed and approved by someone who is not the author of the changes. It is recommended (but not necessary) that a pull request be reviewed by someone who has expertise in the area where the changes are proposed.
- Some automated triggers, like tests, can be integrated with pull requests. That is, a pull request might automatically prompt an automated set of tests to run on the changed code, indicating whether it passes some basic safety checks. Other such checks might include code quality, code linting, or code style checks. The results of such checks are recommended to be logged by the change management system.
- A similar approach is recommended in case of using a ticketing system, it should be ensured that every ticket has an owner who raises an approval request to the relevant head
- Before approving and merging a pull request or ticket, the reviewer checks that all prerequisites are met. Typical checks that the reviewer is encouraged to perform are listed below. Not all items in the list are necessary (depending on the type of change), nor is the list exhaustive. Please use your judgment to determine what is necessary, depending on the change at hand. Below are some questions to ask:
    - Does the proposed change solve the problem it set out to solve? Are all requirements for solving the problem met? If not, were reasonable trade-offs made?
    - Are there any unintended consequences of this change to other parts of the system?
    - Does the change adversely affect any related or unrelated user experience?
    - Are there any algorithmic or logical errors in the proposed change?
    - Does the proposed change require changes in the environment itself (like adding production environment variables etc)?
    - Could the change create performance issues for itself (or other parts of the system)?
    - Could the proposed change be achieved in a more extensible, robust, or less disruptive way?

3.1.4. Unplanned Changes

- Sometimes, it becomes necessary to apply unplanned changes, like hotfixes, to the production system in order to maintain CNTXT's operational effectiveness. This is usually done to address a situation where the production system is in an undesired state - either from a customer-experience standpoint (like critical bugs, system-down, etc.) or from a security standpoint.
- Depending on the urgency of the fix required, unplanned changes may skip the requirements of a peer review/approval. Such requests are peer-reviewed post facto.
- In such cases, we can create changes in a new branch. For all such cases, the commit and/or pull request messages should detail the nature of the issue being fixed as a result of this change.

Unplanned changes follow the same process as planned changes (at least one review or approval from someone who is not the author of the change).

- Since we use a version control system, emergency changes can be rolled back if it has unintended or undesirable consequences.

3.2. Backup Procedure

- CNTXT shall have a daily full backup configured for all customer data on the Infrastructure operated on CNTXT.
- The backup retention period shall be configured to a minimum of 7 days.
- Restoration shall be performed on the backup to ensure that data is readable/accessible. This exercise shall be performed at least once every year.
- The restoration tests shall be documented. The backup snapshot that was restored shall be documented along with any sanity checks performed to ensure that the restoration was successful.
- Restoration tests shall be performed by the administrators of CNTXT's production infrastructure along with the Information Security Officer.
- In an unlikely event of a natural or human-induced disaster, a disaster recovery plan needs to be in place for the systems to recover from the failure and be up and running. This can be achieved in the form of tabletop exercises which must be carried out by the Engineering Head and the Information Security Officer.

3.3. Vulnerability Management Procedure

- CNTXT performs various internal vulnerability scans and package monitoring on a constant basis.
- The Information Security Officer must ensure that CNTXT also performs external vulnerability scans/penetration tests periodically.
- All vulnerabilities detected by vulnerability scanners are tracked together along with their severity.
- It is the responsibility of the Infra operations person to ensure that vulnerabilities are remediated by the engineering team within defined SLAs (Process Config page).
- It is important to track the SLAs for the remediation of vulnerabilities. In case SLA is breached, it is the responsibility of the Information Security Officer along with engineering leads to ensure appropriate actions are taken. E.g. If a vulnerability needs more time to remediate, ensure that the justification for the same is documented.
- Vulnerabilities that are identified through external assessments may also be tracked along with other vulnerabilities for their closure if required.
- The engineering team addresses the reported vulnerabilities and tracks them to resolution. Resolution statuses can include (but are not limited to) the following:
    - Fixed: This means that the reported vulnerability has been fixed via a patch or system changes.

- Inaccurate/Incorrect/False-positive: This means that the reported vulnerability has been thoroughly investigated, but found to be invalid.
- Vulnerable but section unused: This means that the reported vulnerability affects parts of the codebase/system that are not in use, and consequently the vulnerability is no longer a threat.
- Acceptable risk: This means that the reported vulnerability has been analyzed and deemed to not pose any debilitating risk to the system. This is a rare-case scenario, and should only occur when there are extenuating circumstances or extremely high remediation costs.

4. Document Security Classification

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

5. 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, up to and including termination of employment or contractual agreement. The disciplinary action shall depend on the extent, intent, and repercussions of the specific violation.

6. 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.

7. Schedule

This document should be reviewed annually and whenever significant changes occur in the organization.

End of Operations Security Procedure. For version history, please see the next page.

Version history

Version	Log	Date
1 Current	Policy version approved by Youssef Ouyhya	26 Oct, 2024
1	New Policy version Created	26 Oct, 2024