DMS LOGISTICS

SAFE DEVELOPMENT POLICY

Code: POL-TI-002
Revision: 01
Data: 13/03/2023

CONTROLE DE QUALIDADE
APROVADO DMS LOGISTICS
ISO 9001:2015

WWW.DMSLOG.COM

1. INTRODUCTION

The purpose of this Policy is to clarify how DMS LOGISTICS carries out the development of its systems and serve as a guide of good practices to be adopted by analysts, developers and all employees who operate DMS LOGISTICS systems,making the process of designing and using the systems more reliable, auditable, stable and protected against threats, in order to meet the concepts of software quality.

DMS LOGISTICS' secure development methodology is based on theOWASP Secure Coding Guide, Microsoft Secure Development Lifecycle practices, as well as the Software Development Lifecycle (SDLC) SCRUM methodology, and utilizes the threat modeling method in addition to segregation of environments.

2. SCOPE

This Policy applies to all information technology employees, whether contractors, temporary workers, contractors or interns, who are in any way involved in the processes of development of systems and applications of DMS LOGISTICS.

3. GENERAL GUIDELINES OF THIS POLICY

- Systems and applications developed, created or deployed in the DMS LOGISTICS environment must follow a standardized secure development lifecycle set forth in this Policy.
- Systems and applications must receive frequent maintenance and are required to undergo periodic updates to address potential vulnerabilities. These processes should occur at least every thirty (30) days or when there is a need/recommendation to update.
- Every system or application developed must be directed at the OWASP Secure Development Guide and other security recommendations contained in this Policy.
- At DMS LOGISTICS, the development, test and production environments are separate.
- There is separation of duties and access controls between personnel assigned to the dev/test environments and those assigned to the production environment. Different servers, databases and networks are used to ensure that there is no data contamination.

- The data production source must be sanitized prior to use in a development or test environment, and production/test access controls must conform to production standards.
- No data is moved from the test environment to the production. Only the system source code is used.
- All employees involved in the development of systems and applications should receive training to write secure code.
- Access to the source code will be restricted, based on the principle of my privilege.
- For systems/software that store and transmit sensitive information, security controls will be implemented to limit output to the minimum required as defined by the user.
- All newly developed systems and applications, as well as updates or revisions thereof should be fully tested and accepted prior to deployment to the production environment.

4. DEVELOPMENT LIFE CYCLE

ISO 27001:2022 establishes, in its control A.8.25 the use of a secure development life cycle, where rules must be established for the secure development of software, systems and applications.

In this Policy, two modalities of secure development lifecycle will be addressed: the SDL, from Microsoft, where security practices are identified during development, and the SDLC, from the SCRUM methodology, where in fact a process is established, divided into well-defined stages of development.

4.1. Secure Development Lifecycle (SDL)

The secure development lifecycle (SDL) consists of a series of practices that ensure the security requirements of a system, application, or software. The SDL (Security Development Lifecycle) helped developers program more securely, reducing the number and severity of vulnerabilities, while also reducing the cost of development.

For reference, this Policy will be based on the Microsoft SDL model.

4.1.1. Practice 1 – Training

As security is a duty of all, it is important that developers, engineers, analysts and managers understand the principles of security, so that they can build systems, applications, software and products more secure while aiming to meet the needs of the business.

Effective training should complement and reinforce the standards established in safety policies.

4.1.2. Practice 2 – Define Safety Requirements

It is necessary to consider security and privacy as a key aspect in development, and regardless of the methodology used in this process, security requirements must be continuously updated to reflect changes in the threat landscape.

The ideal step in defining security requisitos is during the initial design and planning stages, as this allows development teams to integrate security throughout the process and helps minimize disruptions due to security issues.

Some factors that influence the definition of the security requirements of an application, system or software are legal (e.g. the General Data Protection Act) and industrial requirements, internal standards and development practices, review of previous incidents and known threats.

4.1.3. Practice 3 – Define Metrics and Compliance Reporting

It is essential to define the minimum acceptable security quality levels and keep development teams abreast of these levels in order to achieve them.

DDefining this early in the process helps the development team understand the risks associated with security issues, identify and correct security defects during development, and apply safety standards throughout the project. An Indicator of *bugs* is important to clearly define the severity thresholds of security vulnerabilities (for example, all known vulnerabilities discovered with a "critical" or "important" severity level must be fixed with a specific resolution time).

4.1.4. Practice 4 – Performing Threat Modeling

Threat modeling should be used in environments where there is a significant security risk. Threat modeling can be applied in a component,application, or at the system level. It is a practice that allows development teams to consider, document, and discuss the security implications of designs in the context of the planned operating environment in a structured way.

Applyingstructured embroidery to threat scenarios helps the team identify vulnerabilities more effectively and less costly, determining risks from those threats and then establishing appropriate mitigation means. Threat modeling will be better covered in topic 5 (five).

4.1.5. Practice 5 – Establish Design Requirements

The SDL method of development lifecycle is typically seen as a way to ensure activities that help developers implement secure features,in which these features are designed with respect to security. To achieve this, developers will often rely on security features such as encryption, authentication, *logging*, and more. In many cases, the selection or implementation of security features is so complicated that design or implementation choices are likely to end up resulting in vulnerabilities. Therefore, it is crucial that these resources are applied consistently and with a constant understanding of the routing they provide.

4.1.6. Practice 6 – Define and Use Encryption Standards

With the rise of computing on mobile devices and cloud, it is of paramount importance to ensure that all data, including sensitive information, is protected from unintended leakage or alteration when it is being transmitted or stored. Encryption is typically the means used to achieve this goal. Making an incorrect choice in the use of encryption in any respect can be catastrophic, and it is best to develop clear encryption standards that provide specifications regarding all elements in the implementation of encryption.

A good rule of thumb is to only use libraries that are respected and recognized in the industry and ensure that they are implemented in a way that allows them to be easily replaced if the need arises.

4.1.7. Practice 7 – Managing Security Risks in Using Third-Party Components

Currently, the vast majority of software projects are made using third-party components. When selecting third-party components, it is important to understand the impact that a security vulnerability in them can have on the security of the system in which this component will be integrated. Having an accurate inventory of all third-party components and a plan to respond in the event that new vulnerabilities are discovered is of paramount importance to help mitigate potential risks, but additional validations must be considered, such as the type of component used and the potential impact of a vulnerability.

4.1.8. Practice 8 – Used Approved Tools

It is important to define and publish a list of approved tools and the security checks associated with them, such as *compiler/linker* options and warnings. Developers should strive to use the latest versions of these approved tools, as well as be aware of new security reviews, features and means of protection.

4.1.9. Practice 9 – Perform Static Analysis Security Tests (SAST)

Analyzing source code before compiling provides a highly scalable method of reviewing security in code and helps ensure that development security policies are being followed. SAST is typically integrated into the *commit pipeline* to identify vulnerabilities each time software is built. However, some offerings integrate into the developer environment to find certain flaws such as the existence of unsafe or banned functions and replace them with alternatives more certain as the developer is writing the code. There is no one-size-fits-all solution and development teams must decide what is the best periodicity to perform SAST and perhaps use multiple tactics to balance production with adequate security coverage.

4.1.10. Practice 10 – Perform Dynamic Analysis Security Tests (DAST)

Performing a run-time check on your fully compiled or *packaged* software/system/application checks for functionality that is only apparent when all components are integrated and running. This is usually achieved using a preconfigured tool or set of attacks, or even tools that specifically monitor application behavior with memory corruption, user privilege issues, and other critical security issues. It is similar to SAST, there is no one-size-fits-all solution, and while some tools, such as *web scan* tools, can be readily integrated, or other DAST tests, such as *fuzzing*, require a different approach.

4.1.11. Practice 11 – Perform Penetration Testing

Penetration testing (PenTest) is a security analysis of a software/system/application performed by capable security professionals, simulating the actions of a *hacker*. The purpose of a PenTest is to discover potential vulnerabilities, resulting from errors in the code, system configuration errors or other operational vulnerabilities, among other various types of vulnerabilities. PenTests are commonly performed with a set of tools and code reviews both manual and automated, to provide a deep level of analysis.

4.1.12. Practice 12 – Establish a Standard Incident Response Process

Preparing an incident response plan is crucial to assist in addressing new threats that may arise over time. The plan should be created in coordination with the Information Security Committee. This plan should include who should be contacted in the event of a security emergency, and establish the security service protocol, including plans for codes inherited from other groups inside or outside the organization. The incident response plan must be tested before it is really necessary.

4.2. Scrum Methodology-Development Lifecycle (SDLC)

Unlike the SDL, which consists of practices to ensure security in the development of a system, software or application, the SDLC is a model of agile development processes, based on the SCRUM methodology, which divides development into well-organized stages, each with its own goal.

DMS LOGISTICS operates following the following stages of development: Requirements Analysis, Design and Implementation, Code Review, Teste in Homologation Environment and Publication in production.

This whole process is repeated in a cycle, called SPRINTS. At first they are separated which needs will be met in this cycle and each need goes through processes.

Another important point of the SCRUM methodology is the Daily Scrum, where a quick meeting (of 10 to 15 minutes) is held at the beginning of each day, where members inform what was done the day before and what the planning of the current day is.

Below are the steps that regulate the theme. They must be followed by all developers.

4.2.1. Requirements Analysis

Elaboration and definition of the problem that will be solved by understanding what needs will be met with this demand. At this stage, the care regarding the security of information at a conceptual level aims to:

- Ensure that the new demand does not expose users with lower permission levels to third-party information.
- Ensure that the new demand does not expose third-party information to system users from other accounts or non-system users.
- The new demand should not change information provided by system users.

4.2.2. Design and Implementation

Stage where the design of the solution is defined adapting the architecture and technology used and the implementation. At this stage, care regarding information security at a conceptual level is aimed at:

- Develop using unit and functional tests so that features already available are not affected.
- Use secure coding practices to prevent:
  - SQL Injection
  - Cross-site Request Forgery
  - Other threats that may target the code itself
- Design solutions that have minimal impact on previous customer data.

4.2.3. Code Review

At this stage the code and solution produced are reviewed to ensure that the functional and non-functional objectives, which are likely to be detected through the analysis of the code (example: maintainability, availability and performance), are met. At this stage, care regarding information security at a conceptual level is aimed at:

- Ensure that programming best practices have been used.

- Ensure that programming practices prevent SQL Injection attacks, CSRF attacks.
- Ensure that only functionality has been applied, and extra code has not been added.
- Ensuring that libraries or third-party code that have known vulnerabilities are not introduced into the new code.
- Ensuring that the new code does not expose system loopholes to third parties and system users.

4.2.4. Testing in an Approval Environment

At this stage the solution is tested, in order to find faults in the operation, and also other functional requirements are tested, such as performance. At this stage, the care related to information security aims to:

- Ensure that the system and new functionality work.
- Ensure that the system works on different loads. Load tests are done on more critical functionalities.
- Ensure that the new features did not open loopholes for possible attacks.

4.2.5. Publication in Production

Once The development cycle for each of the features, fixes and improvements expected by the sprint has been completed, publication is made in production. At this stage, the care related to information security aims to:

- Ensure that the system remains available.
- The entire publication process is done in an automated way, avoiding risks of omission of execution of steps for publication of the new version of the system.
- The entire publishing process ensures publication without downtime in case we need to make a comment or emergency adjustments.
- The publication of most functionalities is done at a time when it will not

compromise the use by users.

- After publishing, the application is monitored and when unexpected exceptions happen, it is notified to an internal system that manages these exceptions so that they can be corrected.

5. THREAT MODELING

Threat modeling is a technique capable of assisting in the protection of systems, applications, software, networks and services. It works by identifying potential threats and developing risk mitigation strategies in the development lifecycle.

This is done using a data flow diagram, graphically showing how the whole system works. From this, the modeling applies a structure that assists in the identification and correction of security problems.

Threat modeling should be developed by someone who knows how the system works and has working knowledge about information security.

Currently, the use of threat modeling is recommended by the vast majority of information security entities, including OWASP. Therefore, its implementation in the technology operations model of DMS LOGISTICS is fundamental to ensure a safe development.

Threat modeling is divided into four phases, each of which contains important steps in creating a data flow diagram, allowing you to analyze it for possible threats.

5.1. Step 1 – Design

It aims to capture all the system requirements and important information, and then create the data flow diagram.

5.2. Step 2 – Failure

A threat modeling framework is applied to the data flow diagram to find potential security issues. The diagram is used to help find the most common threats and ways to propagate against them. Examples of the most common threats that can be observed in the framework are: forgery, tampering, repudiation, disclosure of confidential information, denial of service, and elevation of privilege.

5.3. Step 3 – Correction

It is decided how each problem will be addressed. That is, the fate of the threats is decided, mapping each of them to one or more security controls, whether physical, technical or administrative, and defining priority levels for the resolution of each action.

5.4. Step 4 – Verify

It is the last phase of the threat modeling process, where the checks are made, for example, if the security requirements have been met, if problems have been found and if there has been implementation of the controls. It is important to perform manual and automated verification.

6. OWASP SECURE DEVELOPMENT GUIDE

Building secure software requires basic knowledge of security principles. The goal of application security is to maintain the confidentiality, integrity, and availability of information resources in order to enable business operations to be successful, and this goal is achieved through the implementation of security controls.

The security of applications and computer software is one of the most important stages of planning for development. DMS LOGISTICS uses the OWASP methodology to ensure the security of its applications/systems.

OWASP periodically creates a top 10 of the most common vulnerabilities in cybercrime . This list provides us with a guide to top threats and how to protect against them.

The latest update of the top 10 (2021), introduced the vulnerability "Insecure Design", that is, insecure design, demonstrating the need for care that must be taken regarding the development of software/systems/applications.

In addition to this, other vulnerabilities concern development, such as "Injection", (which is a vulnerability in the database, usually when concatenation is allowed, which opens a loophole for injection), among other vulnerabilities that can be avoided with a development that follows security parameters. Below is the updated list of the top 10 vulnerabilities, according to OWASP:

- Broken Access Control
- Cryptographic Failures
- Injection
- Insecure design
- Incorrect Security Configuration
- Vulnerable or Outdated Components

- Identification and Authentication Failures
- Software and Data Integrity Failures
- Security and Monitoring Log Failures
- Server-side Request Forgery (SSRF)

To avoid all these vulnerabilities, OWASP has created and updates with some frequency, according to changes in the threat landscape, the Secure Coding Best Practices Guide, which is used by DMS LOGISTICS, according to the needs and particularities of its systems/software/applications, in order to mitigate possible vulnerabilities.

The purpose of the OWASP Guide is to define and recommend a set of good development safety practices. Recommendations are presented in the form of a checklist. In addition to this list of checks, the Guide provides a list of general best practices:

- Clearly define roles and responsibilities
- Provide development teams with adequate training in development security
- Implement a secure development cycle
- Establish secure programming standards
- Build a reusable library or make use of a security library
- Verify the effectiveness of security controls
- Establish practices to ensure security when there is outsourcing in development, including the definition of security requirements and verification methodologies for both the requirements of the proposals and the contract to be signed between the parties

The Guide also presents general principles of security in applications and risks, with the main aspects that should be taken into account when developing a software/system/application.

To download the Portuguese (Brazil) version of the Guide, use the link: https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/

7. TEST PLAN

To ensure an acceptable level of security in our systems, several tests should be done during the life phase of the application/system/software. Among them: functional tests, integration tests and system tests.

The tests are performed after the implementation of each function or method that will compose the code as a whole.

The integration test will be performed after the integration of two or more functions or methods and the system test will be performed at the end of the implement/coding phase, using the following tests:

- Tests using automated tools in the development environment always at the end of each sprint.
- Annual security testing done by a human to look for vulnerabilities that cannot be detected by automated tools (production environment).

In the production environment three types of tests should be performed:

- Black box
- Gray box
- White box

8. CODE REVIEW

Code review helps developers learn the code base. When The developer finishes working on a problem, another developer analyzes the code and considers questions such as:

- Are there obvious logic errors in the code?
- Looking at the requirements, are all the cases fully implemented?
- Are the new automated tests sufficient for the new code? Do existing automated tests need to be rewritten to justify code changes?
- Does the new code conform to existing style guidelines?

Code reviews should be integrated with the team's existing process, according to the need presented in the specific case.

9. SEGREGATION OF ENVIRONMENTS

DMS LOGISTICS adopts the segregation of environments. This practice increases security in development, test, and production environments. The key principle of segregation of environments is to ensure the integrity and availability (two of the pillars of information security) of the computer environment and the data that resides in it, protect against unauthorized access, changes and other negative impacts.

Separating the operating environment prevents cross-contamination of applications and is a best practice in the systems development lifecycle. In doing so, DMS LOGISTICS seeks to prevent the software from being used in production before it is ready.

By isolating application environments, IT/DevOps teams can help prevent disruptions, errors, and compliance violations that could negatively impact DMS LOGISTICS' revenue or reputation and cause financial penalties.

Different access rules and data integrity considerations apply to development and test environments.

9.1 Objective

The purpose of segregation of environments is:

- Prevent unwanted client access to business-critical servers;
- Perform user acceptance testing of applications prior to deployment;
- Application isolation through lifecycle stages;
- Conducting compliance audits;
- Isolate security vulnerabilities;
- Prevent unwanted developer access to production environments or limited to troubleshooting, with all activities being logged and monitored.
- Ensure security in authentication practices by ensuring that the credentials of each environment are unique and unrelated.
- Define procedures for transferring software or hardware from development and testing to production.

9.2 Environments

Our environments are separated into three: Development, QA and Production.

The production environment is completely isolated from the development and QA environments.

The entire DMS LOGISTICS infrastructure is on Amazon Web Services (AWS).

Within each environment, the computational resources necessary for the operation are provisioned, and they are logically isolated through virtual private networks networks based on IPSEC (IP Security Protocol) of Amazon VPC (Virtual Private Cloud).

External access to internal resources in your VPC is routed through AWS Network Firewall. All external access control to networks is controlled by Amazon and contains security rules for each environment.

10. DATA IN A TEST ENVIRONMENT

The development process involves a series of activities in which it is very common for errors to occur that can happen throughout the process, from the conception to the construction of the application/system/software. The test activity aims to assist in the discovery of these errors, as soon as possible, so that the cost of correcting them is as low as possible.

The test stage is an important stage of the Software Development Lifecycle and can decide the final destination of the quality of the application/system/software being released in DMS LOGISTICS' technology environments. Therefore, it is very important to ensure that the test environments used to test the software are reliable and as close to production as possible.

Testing should take place to ensure that when the system is launched, as many problems as possible are resolved. The test needs to assess the security and robustness of the system and the data it processes.

The use of personal data in test environments is one of the most critical issues of this step. Ensuring the integrity, confidentiality, and availability of data even in test environments is the primary objective of this step and will be addressed in this document.

The specific type of data to be processed or the role of the system may require the use of the production data to properly test its capabilities.

Test environments are often not as complete as production environments, so certain components of a system can only be tested properly in an active environment.

As it may not be possible to replicate a particularly specialized process within the test environment due to limitations of the process itself or the data it requires, to ensure data security within the test environment without jeopardizing effective testing of the software/system/application, some measures are necessary and applied by DMS LOGISTICS.

All personal data used in the tests must be strictly relevant to the purpose of the test. Individual data items should be listed and identified as:

- Non-personal
- Personal
- Sensitive people

Once the data has been classified, the reasons for inclusion in the test should be provided. Where a reason for inclusion cannot be found, personal data should not be used.

The testing method needs to ensure that data integrity is maintained and any risks to accuracy are mitigated.

It is important that a system testing regime is implemented to maintain an audit trail to catch errors during the testing process and enable immediate correction. Integrity checks should be performed on the data being fed into the system, especially if there is a possibility that this data could be merged with other data or fed back into the source system.

Mechanisms should be in place to ensure that test data is not retained for longer than necessary to meet legal, regulatory or commercial requirements. They will be destroyed after the tests. Personal data will not be disclosed for testing purposes to third parties. If this is necessary, authorization will be requested for the data subjects.

10.1. Techniques Used

The tests should be performed using:

- Shuffling of primary keys;
- Attribution of null values where the information is more sensitive;
- Truncate data;
- Letter shuffling and random data generation;
- No production image is used in the test environment;
- Restricted access to the development team, through two-factor authentication.

10.2. Data Masking/Anonymization

Before performing the tests, considerations should be made to identify whether fictitious or anonymized data can be used. If fictitious test data cannot be used, there are several techniques that can be used to effectively mask the data so that risks are mitigated. Some data testing and masking techniques adopted by DMS LOGISTICS are listed below.

10.2.1. Substitution

This technique consists of randomly replacing the contents of a cell or column of data with information that looks similar but is not related to the actual information. For example, real names are replaced by those extracted from a random list.

10.2.2. Shuffle

Shuffling is similar to substitution, but the value of one cell is replaced with data derived from other cells in that column. This method is not as useful in smaller databases, where a lack of variety between values can cause a problem.

10.2.3. Change in Number and Date

This method is useful for numeric and date values. An algorithm is implemented that modifies each value (such as date of birth) by a random percentage. This technique has the advantage of being a reasonable way to mask data while maintaining its usefulness.

10.2.4. Pseudonymization

Pseudonymisation, or "the processing of personal data in such a way that personal data can no longer be attributed to a specific data subject (the person to whom the data relate) without the use of additional information, provided that such additional information is kept separately and subject to technical and organizational measures to ensure that the personal data are not attributed to an identified or identifiable natural person".

10.3. Accountability

DMS LOGISTICS maintains documentation regarding the processing of personal data for testing, identifying how and if the principles of the LGPD have been complied with, what risks exist for the data and how the company mitigated these risks.

Compliance is demonstrated in the following ways:

- Keeping information of the asset being tested;
- Retention of audit logs and access from when the tests occurred.

10.4. Test Plan

To ensure an acceptable level of security in DMS LOGISTICS systems, the following tests should be performed:

- Tests using automated tools in the development environment always at the end of each sprint.
- Annual security testing done by a human to look for vulnerabilities that cannot be detected by automated tools (production environment).

In the production environment two types of tests should be performed:

- Black box
- Gray box

11. ACCESS TO SOURCE CODE AND REPOSITORIES

Access to the source code is restricted, based on the principle of least privilege. The versioning control used by DMS LOGISTICS is Git (Atlassian Bitbucket). Everyone who handles source code or manages/maintains source code repositories must follow the following requirements designed to protect source code from unauthorized access, theft, alteration, and loss.

The source code of DMS LOGISTICS should never be transferred in an unencrypted form.

The passwords and other credentials used to access the source code repositories are stored in an encrypted format by identity management on the Atlassian Bitbucket platform.

It is acceptable to store tokens, cookies, tickets, or other authentication credentials locally, as long as they do not contain unencrypted passwords (or equivalents).

Access to the repositories is done through public key cryptography, of the SSL (Secure Sockets Layer) type, individual to each user in addition to the gin and password of the Atlassian Bitbucket platform. To gain access, it is necessary to request it from the repository administrator.

12. NON-CONFORMITY

In cases where it is determined that a violation of DMS LOGISTICS' policies has occurred, corrective measures will be taken, including restriction of access to services or initiation of disciplinary action, which may result in dismissal.

13. STATEMENT OF COMMITMENT

When there are new Officers, Employees, Service Providers and Partners of DMS LOGISTICS, they undertake to follow and implement the DMS LOGISTICS Policies.

14. UPDATE

The DMS LOGISTICS Desk and Clean Screen Policy must be updated whenever necessary or at an interval not exceeding one (1) year.

15. REVISION HISTORY

Revision	Data	Description
00	13/02/2023	Issuance of the document.
01	10/03/2023	Document review and standardization

16. APPROVAL AND CLASSIFICATION OF INFORMATION

Prepared by:	CyberSecurity Team	
Reviewed by:	Leonardo Sabbadim	
Approved by:	Victor Gonzaga	
Level of Confidentiality:	X	Public Information
		Internal Information
		Confidential Information
		Confidential Information

DMS LOGISTICS

WE NEVER PUT QUALITY OR ETHICS AT RISK IN BUSINESS

WE NEVER COMPROMISE ON QUALITY AND BUSINESS ETHICS

WWW.DMSLOG.COM