POLÍTICA DE DESENVOLVIMENTO SEGURO

Código: POL-TI-002
Revisão: 01
Data: 13/03/2023

WWW.DMSLOG.COM

1. INTRODUÇÃO

O objetivo desta Política é esclarecer como a DMS LOGISTICS realiza o desenvolvimento de seus sistemas e servir como guia de boas práticas a serem adotadas por analistas, desenvolvedores e todos os colaboradores que operam os sistemas da DMS LOGISTICS, tornando o processo de concepção e uso dos sistemas mais confiável, auditável, estável e protegido contra ameaças, a fim de atender aos conceitos de qualidade de software.

A metodologia de desenvolvimento seguro da DMS LOGISTICS se baseia no Guia de Codificação Segura da OWASP, nas práticas do Ciclo de Vida de Desenvolvimento Seguro da Microsoft, bem como na metodologia SCRUM de ciclo de vida de desenvolvimento de softwares (SDLC) e utiliza o método de modelagem de ameaças, além da segregação de ambientes.

2. ESCOPO

Esta Política se aplica a todos os colaboradores de tecnologia da informação, sejam contratados, trabalhadores temporários, terceirizados ou estagiários, que estejam de algum modo envolvidos nos processos de desenvolvimento de sistemas e aplicações da DMS LOGISTICS.

3. DIRETRIZES GERAIS DESSA POLÍTICA

- Os sistemas e aplicações desenvolvidos, criados ou implantados no ambiente da DMS LOGISTICS devem seguir um ciclo de vida de desenvolvimento seguro padronizado, estabelecido nesta Política.
- Os sistemas e aplicações devem receber manutenção frequente e é exigido que passem por atualizações periódicas para lidar com possíveis vulnerabilidades. Esses processos deverão ocorrer no mínimo a cada 30 (trinta) dias ou quando existir a necessidade/recomendação de atualização.
- Todo sistema ou aplicação desenvolvido deve ter como direcionamento o Guia de Desenvolvimento Seguro da OWASP e demais recomendações de segurança contidas nesta Política.
- Na DMS LOGISTICS, os ambientes de desenvolvimento, teste e produção são separados.
- Existe separação de funções e controles de acesso entre o pessoal designado para os ambientes de desenvolvimento/teste e aqueles designados para o ambiente de produção. São utilizados servidores, bases de dados e redes diferentes para garantir que não haja contaminação de dados.

- A fonte de produção de dados deve ser higienizada antes do uso em ambiente de desenvolvimento ou teste e os controles de acesso de produção/teste devem estar em conformidade com os padrões de produção.
- Nenhum dado é movimentado do ambiente de testes para produção. Apenas o código fonte do sistema é utilizado.
- Todos os funcionários envolvidos no desenvolvimento de sistemas e aplicações devem receber treinamento para escrever códigos seguros.
- O acesso ao código fonte será restrito, com base no princípio do menor privilégio.
- Para sistemas/softwares que armazenam e transmitem informações confidenciais, serão implementados controles de segurança para limitar o output ao mínimo necessário, conforme definido pelo usuário.
- Todos os sistemas e aplicações recém-desenvolvidos, bem como atualizações ou revisões dos mesmos deverão ser totalmente testados e aceitos antes da implantação no ambiente de produção.

4. CICLO DE VIDA DE DESENVOLVIMENTO

A ISO 27001:2022 estabelece, em seu controle A.8.25 o uso de um ciclo de vida de desenvolvimento seguro, onde devem ser estabelecidas regras para o desenvolvimento seguro de software, sistemas e aplicações.

Nesta Política, serão abordadas duas modalidades de ciclo de vida de desenvolvimento seguro: o SDL, da Microsoft, onde são estabelecidas práticas de segurança durante o desenvolvimento e o SDLC, da metodologia SCRUM, onde de fato é estabelecido um processo, dividido em etapas bem definidas de desenvolvimento.

4.1. CICLO DE VIDA DE DESENVOLVIMENTO SEGURO (SDL)

O ciclo de vida de desenvolvimento seguro (SDL) consiste em uma série de práticas que assegurem os requisitos de segurança de um sistema, aplicação ou software. O SDL (Security Development Lifecycle) auxilia desenvolvedores a programar com mais segurança, reduzindo o número e a gravidade das vulnerabilidades, enquanto reduz também o custo do desenvolvimento.

Para fins de referência, esta Política utilizará como base o modelo de SDL da Microsoft.

4.1.1. PRÁTICA 1 – TREINAMENTO

Como a segurança é um dever de todos, é importante que desenvolvedores, engenheiros, analistas e gestores entendam os princípios de segurança, para que possam construir sistemas, aplicações, softwares e produtos mais seguros enquanto ainda visam atender as necessidades do negócio.

Um treinamento efetivo deve complementar e reforçar as normas estabelecidas nas políticas de segurança.

4.1.2. PRÁTICA 2 – DEFINIR OS REQUISITOS DE SEGURANÇA

É necessário considerar a segurança e a privacidade como um aspecto fundamental no desenvolvimento e, independentemente da metodologia utilizada neste processo, os requisitos de segurança devem ser continuamente atualizados para refletir as mudanças no cenário de ameaças.

A etapa ideal para definir os requisitos de segurança é durante o design inicial e estágios de planejamento, já que isso permite que os times de desenvolvimento integrem a segurança durante todo o processo e contribui para minimizar interrupções por problemas de segurança.

Alguns fatores que influenciam na definição dos requisitos de segurança de uma aplicação, sistema ou softwares são os requisitos legais (por exemplo, a Lei Geral de Proteção de Dados) e industriais, padrões internos e práticas de desenvolvimento, revisão de incidentes anteriores e ameaças conhecidas.

4.1.3. PRÁTICA 3 – DEFINIR MÉTRICAS E RELATÓRIO DE CONFORMIDADE

É essencial definir os níveis de qualidade de segurança mínimos aceitáveis e manter os times de desenvolvimento a par desses níveis, de modo a atingi-los.

Definir isso logo no início do processo ajuda o time de desenvolvimento a entender os riscos associados a problemas de segurança, identificar e corrigir defeitos de segurança durante o desenvolvimento e aplicar os padrões de segurança em todo o projeto. Estabelecer um indicador de *bugs* (erros) é importante para definir claramente os limiares de gravidade das vulnerabilidades de segurança (por exemplo, todas as vulnerabilidades conhecidas descobertas com um nível de gravidade “crítico” ou “importante” devem ser fixadas com um tempo de resolução específico).

4.1.4.PRÁTICA 4 – PERFORMAR A MODELAGEM DE AMEAÇAS

A modelagem de ameaças deve ser usada em ambientes onde existe um risco de segurança significativo. A modelagem de ameaças pode ser aplicada em um componente, aplicação ou a nível de sistema. É uma prática que permite que os

times de desenvolvimento considerem, documentem e discutam as implicações de segurança dos designs no contexto do ambiente operacional planejado, de um modo estruturado.

Aplicar uma abordagem estruturada em cenários de ameaça ajuda o time a identificar vulnerabilidades de modo mais efetivo e menos custoso, determinando riscos oriundos dessas ameaças para então estabelecer meios de mitigação apropriados. A modelagem de ameaças será melhor abordada no tópico 5 (cinco).

4.1.5. PRÁTICA 5 – ESTABELECER REQUISITOS DE DESIGN

O método SDL de ciclo de vida de desenvolvimento é tipicamente visto como uma forma de assegurar atividades que ajudem desenvolvedores a implementar recursos seguros, em que esses recursos são projetados com respeito à segurança. Para alcançar isso, os desenvolvedores geralmente vão se basear em recursos de segurança como criptografia, autenticação, *logging* (registro), entre outros. Em muitos casos, a seleção ou implementação de recursos de segurança é tão complicada que as escolhas do design ou implementação provavelmente vão acabar resultando em vulnerabilidades. Por isso, é crucial que esses recursos sejam aplicados de modo consistente e com um constante entendimento da proteção que eles fornecem.

4.1.6.PRÁTICA 6 – DEFINIR E UTILIZAR PADRÕES DE CRIPTOGRAFIA

Com o aumento de computação em dispositivos móveis e cloud, é de suma importância garantir que todos os dados, incluindo as informações sigilosas estão protegidos de vazamento não intencionais ou alteração quando estão sendo transmitidos ou armazenados. Encriptação tipicamente é o meio utilizado para alcança esse objetivo. Fazer uma escolha incorreta no uso de criptografia em qualquer aspecto pode ser catastrófico, e é melhor desenvolver padrões claros de encriptação que forneçam especificações a respeito de todos os elementos na implementação da encriptação.

Uma boa regra geral é somente usar bibliotecas que são respeitas e reconhecidas na indústria e garantir que elas são implementadas de uma maneira que permita que sejam facilmente substituídas, caso haja necessidade.

4.1.7.PRÁTICA 7 – GERENCIAR OS RISCOS DE SEGURANÇA EM UTILIZAR COMPONENTES TERCEIRIZADOS

Atualmente, a vasta maioria dos projetos de software são feitos utilizado componentes terceirizados. Quando se selecionam componentes terceirizados, é

importante entender o impacto que uma vulnerabilidade de segurança nos mesmos pode ter na segurança do sistema em que esse componente será integrado. Possuir um inventário preciso de todos os componentes terceirizados e um plano para responder no caso de novas vulnerabilidades serem descobertas é de suma importância para ajudar a mitigar possíveis riscos, mas validações adicionais devem ser consideradas, como o tipo de componente utilizado e o potencial impacto de uma vulnerabilidade.

4.1.8. PRÁTICA 8 – UTILIZAR FERRAMENTAS APROVADAS

É importante definir e publicar uma lista de ferramentas aprovadas e as checagens de segurança associadas a elas, como as opções de *compiler/linker* e avisos. Desenvolvedores devem se esforçar para utilizar as versões mais recentes dessas ferramentas aprovadas, além de estarem a par de novas análises de segurança, funcionalidades e meios de proteção.

4.1.9. PRÁTICA 9 – PERFORMAR TESTES DE SEGURANÇA DE ANÁLISE ESTÁTICA (SAST)

Analisar o código fonte antes de compilar fornece um método altamente escalável de revisão de segurança no código e ajuda a garantir que as políticas de segurança de desenvolvimento estão sendo seguidas. O SAST é geralmente integrado no *pipeline* do *commit* para identificar vulnerabilidades cada vez que o software é construído. Porém, algumas ofertas se integram no ambiente do desenvolvedor para encontrar certas falhas como a existência de funções não seguras ou banidas e as substituem por alternativas mais seguras conforme o desenvolvedor está escrevendo o código. Não existe uma solução que serve para tudo e times de desenvolvimento devem decidir qual é a melhor periodicidade para performar o SAST e talvez utilizar múltiplas táticas, para balancear produtividade com cobertura de segurança adequada.

4.1.10. PRÁTICA 10 – PERFORMAR TESTES DE SEGURANÇA DE ANÁLISE DINÂMICA (DAST)

Performar uma verificação em tempo de execução em seu software/sistema/aplicação totalmente compilado ou *packaged* checa a funcionalidade que só é aparente quando todos os componentes estão integrados e rodando. Isso geralmente é atingido utilizando uma ferramenta ou conjunto de ataques pré-configurados, ou mesmo ferramentas que monitoram especificamente o comportamento da aplicação com corrupção da memória, problemas de privilégio de usuário e outros problemas críticos de segurança. É similar ao SAST, não existe uma solução que serve para tudo e, enquanto algumas ferramentas, como

ferramentas de *web scan*, podem ser prontamente integradas, outros testes DAST, como *fuzzing*, requerem uma abordagem diferente.

4.1.11. PRÁTICA 11 – PERFORMAR TESTES DE PENETRAÇÃO

Teste de penetração (PenTest) é uma análise de segurança de um software/sistema/aplicação performado por profissionais de segurança capacitados, simulando as ações de um *hacker*. O objetivo de um PenTest é descobrir potenciais vulnerabilidades, resultantes de erros no código, erros de configuração do sistema ou outras vulnerabilidades operacionais, dentre outros diversos tipos de vulnerabilidades. PenTests são comumente performados com um conjunto de ferramentas e revisões no código tanto manuais como automatizadas, para prover um nível profundo de análise.

4.1.12. PRÁTICA 12 – ESTABELECER UM PADRÃO DE PROCESSO DE RESPOSTA A INCIDENTES

Preparar um plano de resposta a incidentes é crucial para auxiliar no endereçamento de novas ameaças que podem surgir ao longo do tempo. O plano deve ser criado em coordenação com Comitê de Segurança da Informação. Esse plano deve incluir quem deve ser contatado em caso de uma emergência de segurança, e estabelecer o protocolo de serviço de segurança, incluindo planos para códigos herdados de outros grupos dentro ou fora da organização. O plano de resposta a incidentes deve ser testado antes que seja realmente necessário.

4.2. METODOLOGIA SCRUM-CICLO DE VIDA DE DESENVOLVIMENTO (SDLC)

Diferente do SDL, que consiste em práticas para assegurar a segurança no desenvolvimento de um sistema, software ou aplicação, o SDLC é um modelo de processos de desenvolvimento ágil, baseado na metodologia SCRUM, que divide o desenvolvimento em etapas bem organizadas, cada uma com o seu objetivo.

A DMS LOGISTICS opera seguindo as seguintes etapas de desenvolvimento: Análise de Requisitos, Projeto e Implementação, Revisão de Código, Teste em Ambiente de Homologação e Publicação em produção.

Todo esse processo se repete num ciclo, chamado de SPRINTS. Em um primeiro momento são separadas quais as necessidades serão atendidas neste ciclo e cada necessidade passa por processos.

Outro ponto importante da metodologia SCRUM é o Daily Scrum, onde é feita uma reunião rápida (de 10 a 15 minutos) no início de cada dia, onde os membros informam o que foi feito no dia anterior e qual o planejamento do dia atual.

Seguem, abaixo, as etapas que regulam o tema. Elas devem ser seguidas por todos os desenvolvedores.

4.2.1. ANÁLISE DE REQUISITOS

Elaboração e definição do problema que será resolvido compreendendo quais necessidades serão atendidas com essa demanda. Nesta etapa os cuidados referentes a segurança da informação num nível conceitual visam:

- Garantir que a nova demanda não expõe usuários com níveis de permissões mais baixos a informações de terceiros.
- Garantir que a nova demanda não expõe informações de terceiros para usuários do sistema de outras contas e nem usuários que não são do sistema.
- A nova demanda não deverá alterar informações fornecidas por usuários do sistema.

4.2.2. PROJETO E IMPLEMENTAÇÃO

Etapa onde é definido o projeto da solução adequando a arquitetura e tecnologia usadas e a implementação. Nesta etapa os cuidados referentes a segurança da informação num nível conceitual visam:

- Desenvolver utilizando testes unitários e funcionais para que funcionalidades já disponíveis não sejam afetadas.
- Utilizar práticas de codificação segura para prevenir:
  - SQL Injection
  - Cross-site Request Forgery
  - Outras ameaças que possam atingir o código em si
- Projetar soluções que tenham o mínimo impacto em dados anteriores de clientes.

4.2.3. REVISÃO DE CÓDIGO

Nessa etapa o código e solução produzidos são revisados para garantir que os objetivos funcionais e não funcionais, que são passíveis de detectar através da análise do código (exemplo: manutenibilidade, disponibilidade e performance), necessários sejam atendidos. Nesta etapa os cuidados referentes a segurança da informação num nível conceitual visam:

- Garantir que as melhores práticas de programação foram usadas.

- Garantir que as práticas de programação previnam ataques de SQL Injection, ataques de CSRF.
- Garantir que somente a funcionalidade foi aplicada, e códigos extras não foram adicionados.
- Garantia de que no código novo não são introduzidas bibliotecas ou códigos de terceiros que estejam com vulnerabilidades conhecidas.
- Garantia de que o código novo não expõe brechas ao sistema a terceiros e a usuários do sistema.

4.2.4.TESTE EM AMBIENTE DE HOMOLOGAÇÃO

Nessa etapa a solução é testada, com objetivo de encontrar falhas no funcionamento, e também outros requisitos funcionais são testados, como desempenho. Nesta etapa os cuidados referentes a segurança da informação visam:

- Garantir que o sistema e a nova funcionalidade funcionem.
- Garantir que o sistema funcione em diferentes cargas. São feitos testes de carga em funcionalidades mais críticas.
- Garantir que as novas funcionalidades não abriram brechas para eventuais ataques.

4.2.5. PUBLICAÇÃO EM PRODUÇÃO

Completado o ciclo de desenvolvimento para cada uma das funcionalidades, correções e melhorias esperadas pelo sprint, é feita a publicação em produção. Nesta etapa os cuidados referentes a segurança da informação visam:

- Garantir que o sistema continue disponível.
- Todo o processo de publicação é feito de forma automatizada, evitando riscos de omissão de execução de etapas para publicação da nova versão do sistema.
- Todo o processo de publicação garante a publicação sem downtime, caso precisemos de fazer uma correção ou ajustes de emergência.
- A publicação de grande parte de funcionalidades é feita em horário em que não comprometerá a utilização pelos usuários.
- Após a publicação, a aplicação é monitorada e quando exceções inesperadas acontecem, é notificado a um sistema interno que gerencia essas exceções para que possam ser corrigidas.

5. MODELAGEM DE AMEAÇAS (THREAT MODELING)

A modelagem de ameaças é uma técnica capaz de auxiliar na proteção de sistemas, aplicativos, softwares, redes e serviços. Ela funciona identificando possíveis ameaças e desenvolvendo estratégias de mitigação de risco no ciclo de vida de desenvolvimento.

Isso é feito utilizando um diagrama de fluxo de dados, mostrando graficamente como todo o sistema funciona. A partir disso, a modelagem aplica uma estrutura que auxilia na identificação e correção de problemas de segurança.

A modelagem de ameaças deve ser desenvolvida por alguém que saiba como o sistema funciona e tenha conhecimento prático sobre segurança da informação.

Atualmente, o uso da modelagem de ameaças é recomendado pela grande maioria das entidades de segurança da informação, inclusive a OWASP. Por isso, sua implementação no modelo de operações de tecnologia da DMS LOGISTICS é fundamental para garantir um desenvolvimento seguro.

A modelagem de ameaças é dividida em quatro fases, sendo que cada uma contém etapas importantes na criação de um diagrama de fluxo de dados, permitindo então, analisá-lo quanto a possíveis ameaças.

5.1. ETAPA 1 – DESIGN

Tem como objetivo capturar todos os requisitos do sistema e informações importantes, para então criar o diagrama de fluxo de dados.

5.2. ETAPA 2 – FALHA

Aplica-se uma estrutura de modelagem de ameaças no diagrama de fluxo de dados para encontrar possíveis problemas de segurança. O diagrama é utilizado para ajudar a encontrar as ameaças mais comuns e formas de se proteger contra as mesmas. Exemplos de ameaças mais comuns que podem ser observadas na estrutura são: falsificação, adulteração, repúdio, divulgação de informações confidenciais, negação de serviço e elevação de privilégio.

5.3.ETAPA 3 – CORREÇÃO

Decide-se de que forma será abordado cada problema. Ou seja, o destino das ameaças é decidido, mapeando cada uma delas para um ou mais controles de segurança, sejam eles físicos, técnicos ou administrativos, e definindo níveis de prioridade para a resolução de cada ameaça.

5.4. ETAPA 4 – VERIFICAR

É a última fase do processo de modelagem de ameaças, onde são feitas as verificações, por exemplo, se os requisitos de segurança foram cumpridos, se foram encontrados problemas e se houve implementação dos controles. É importante realizar a verificação manual e automatizada.

6. GUIA DE DESENVOLVIMENTO SEGURO OWASP

Construir software seguro exige o conhecimento básico dos princípios de segurança. O objetivo da segurança em aplicações é manter a confidencialidade, integridade e disponibilidade dos recursos de informação a fim de permitir que as operações de negócios sejam bem sucedidas e esse objetivo é alcançado através da implementação de controles de segurança.

A segurança de aplicativos e softwares de computadores é uma das etapas mais importantes do planejamento para o desenvolvimento. A DMS LOGISTICS utiliza a metodologia OWASP para garantir a segurança das suas aplicações/sistemas.

A OWASP cria, periodicamente, um top 10 de vulnerabilidades mais comuns no meio dos crimes cibernéticos. Esta lista, nos fornece um guia de principais ameaças e como se proteger contra as mesmas.

A última atualização do top 10 (2021), introduziu a vulnerabilidade “Insecure Design”, ou seja, design inseguro, demonstrando a necessidade de cuidados que deve ser tomados quanto ao desenvolvimento de softwares/sistemas/aplicações.

Além desta, outras vulnerabilidades dizem respeito ao desenvolvimento, como “Injection”, (que é uma vulnerabilidade no banco de dados, normalmente quando se permite a concatenação, o que abre brecha para a injeção), dentre outras vulnerabilidades que podem ser evitadas com um desenvolvimento que segue parâmetros de segurança. Abaixo, segue a lista atualizada do top 10 vulnerabilidades, segundo a OWASP:

- Controle de Acesso Quebrado
- Falhas Criptográficas
- Injeção
- Design Inseguro
- Configuração Incorreta de Segurança
- Componentes Vulneráveis ou Desatualizados
- Falhas de Identificação e Autenticação
- Falhas na Integridade de Software e Dados

- Falhas de Registros de Segurança e Monitoramento
- Server-side Request Forgery (SSRF)

Para evitar todas essas vulnerabilidades, a OWASP criou e atualiza com certa frequência, de acordo com as mudanças no cenário de ameaças, o Guia de Melhores Práticas de Codificação Segura, o qual é utilizado pela DMS LOGISTICS, de acordo com as necessidades e particularidades de seus sistemas/softwares/aplicações, a fim de mitigar possíveis vulnerabilidades.

O Guia da OWASP tem o propósito de definir e recomendar um conjunto de boas práticas de segurança no desenvolvimento. As recomendações são apresentadas no formato de uma lista de verificações. Além dessa lista de verificações, o Guia fornece uma lista de práticas gerais recomendadas:

- Definir claramente os papéis e responsabilidades
- Fornecer às equipes de desenvolvimento formação adequada em segurança no desenvolvimento
- Implementar um ciclo de desenvolvimento seguro
- Estabelecer padrões de programação segura
- Construir uma biblioteca reutilizável ou fazer uso de uma biblioteca de segurança
- Verificar a efetividade dos controles de segurança
- Estabelecer práticas para garantir a segurança quando há terceirização no desenvolvimento, incluindo a definição dos requisitos de segurança e metodologias de verificação tanto para os requisitos das propostas, como para o contrato a ser firmado entre as partes

O Guia também apresenta princípios gerais de segurança em aplicações e riscos, com os principais aspectos que devem ser levados em conta no momento do desenvolvimento de um software/sistema/aplicação.

Para realizar o download da versão em Português (Brasil) do Guia, utilize o link: https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/

7. PLANO DE TESTES

Para garantir um nível aceitável de segurança em nossos sistemas, deverão ser feitos diversos testes durante a fase de vida do aplicativo/sistema/software. Entre eles: testes funcionais, testes de integração e testes de sistema.

Os testes funcionais são realizados após a implementação de cada função ou método que irá compor o código como todo.

O teste de integração será realizado após a integração de duas ou mais funções ou métodos e o teste de sistema será realizado ao término da fase de implementação/codificação, utilizando os seguintes testes:

- Testes utilizando ferramentas automatizadas no ambiente de desenvolvimento sempre no fim de cada sprint.
- Testes de segurança anual feito por um humano para buscar vulnerabilidades que não possam ser detectadas por ferramentas automatizadas (ambiente de produção).

No ambiente de produção devem ser realizados três tipos de testes:

- Black box
- Gray box
- White box

8. REVISÃO DE CÓDIGOS

A revisão de código ajuda os desenvolvedores a aprenderem a base de código. Quando o desenvolvedor terminar de trabalhar em um problema, outro desenvolvedor analisa o código e considera perguntas como:

- Há erros de lógica óbvios no código?
- Olhando para os requisitos, todos os casos estão totalmente implementados?
- Os novos testes automatizados são suficientes para o novo código? Testes automatizados existentes precisam ser reescritos para justificar as alterações no código?
- O novo código está em conformidade com as diretrizes de estilo existentes?

As revisões de código devem ser integradas com o processo existente da equipe, de acordo com a necessidade apresentada no caso concreto.

9. SEGREGAÇÃO DE AMBIENTES

A DMS LOGISTICS adota a segregação de ambientes. Esta prática aumenta a segurança nos ambientes de desenvolvimento, teste e produção. O princípio-chave da segregação de ambientes é assegurar a integridade e disponibilidade (dois dos pilares da segurança da informação) do ambiente de computação e os dados que nele residem, proteger contra acesso não-autorizado, alterações e outros impactos negativos.

A separação do ambiente operacional evita a contaminação cruzada de aplicativos e é uma prática recomendada no ciclo de vida de desenvolvimento de sistemas. Ao fazer isso, a DMS LOGISTICS busca impedir que o software seja usado na produção antes de estar pronto.

Ao isolar ambientes de aplicativos, as equipes de TI/DevOps podem ajudar a evitar interrupções, erros e violações de conformidade que podem afetar negativamente a receita ou a reputação da DMS LOGISTICS e causar penalidades financeiras.

Diferentes regras de acesso e considerações de integridade de dados se aplicam aos ambientes de desenvolvimento e teste.

9.1 OBJETIVO

O objetivo da segregação de ambientes é:

- Evitar o acesso indesejado de clientes a servidores críticos para os negócios;
- Realizar o teste de aceitação do usuário de aplicativos antes da implantação;
- Isolamento de aplicativos através dos estágios do ciclo de vida;
- Realização de auditorias de conformidade;
- Isolar vulnerabilidades de segurança;
- Impedir o acesso indesejado do desenvolvedor aos ambientes de produção ou limitado à solução de problemas, sendo todas as atividades registradas e monitoradas.
- Garantir a segurança nos processos de autenticação ao garantir que as credenciais de cada ambiente sejam únicas e não relacionadas.
- Definir procedimentos para a transferência de software ou hardware do desenvolvimento e teste para a produção.

9.2 AMBIENTES

Nossos ambientes são separados em três: Desenvolvimento, QA e Produção.

O ambiente de produção é completamente isolado dos ambientes de desenvolvimento e QA.

Toda a infraestrutura da DMS LOGISTICS está na Amazon Web Services (AWS).

Dentro de cada ambiente são provisionados os recursos computacionais necessários para a operação, sendo eles isolados logicamente através de redes virtuais privadas baseadas em IPSEC (Protocolo de Segurança IP) do Amazon VPC (Virtual Private Cloud).

Os acessos externos aos recursos internos ao VPC são direcionados pelo AWS Network Firewall. Todo o controle de acesso externo às redes é controlado pela Amazon e contém regras de segurança para cada ambiente.

10. DADOS EM AMBIENTE DE TESTE

O processo de desenvolvimento envolve uma série de atividades nas quais é muito comum a ocorrência de erros que podem acontecer durante todo o processo, desde a concepção até a construção da aplicação/sistema/software. A atividade de teste se propõe a auxiliar na descoberta destes erros, o quanto antes, para que o custo da correção dos mesmos seja o menor possível.

A etapa de teste é um estágio importante do Ciclo de Vida do Desenvolvimento de Softwares e pode decidir o destino final da qualidade da aplicação/sistema/software que está sendo lançado nos ambientes de tecnologia da DMS LOGISTICS. Portanto, é muito importante garantir que os ambientes de teste usados para testar o software sejam confiáveis e o mais próximo possível da produção.

Devem ocorrer os testes para garantir que, quando o sistema for lançado, o maior número possível de problemas esteja resolvido. O teste precisa avaliar a segurança e a robustez do sistema e dos dados que ele processa.

O uso de dados pessoais em ambientes de teste constitui um dos assuntos mais críticos dessa etapa. A garantia da integridade, confidencialidade e disponibilidade dos dados mesmo em ambientes de teste é o objetivo principal desta etapa e será abordado neste documento.

O tipo específico de dados a serem processados ou a função do sistema podem exigir o uso dos dados de produção para testar adequadamente seus recursos.

Ambientes de teste muitas vezes não são tão completos quanto os ambientes de produção, então certos componentes de um sistema só podem ser testados adequadamente em um ambiente ativo.

Como pode não ser possível replicar um processo particularmente especializado dentro do ambiente de teste devido a limitações do próprio processo ou dos dados que ele requer, para garantir a segurança dos dados dentro do ambiente de testes sem prejudicar uma efetiva testagem do software/sistema/aplicação, algumas medidas são necessárias e aplicadas pela DMS LOGISTICS.

Todos os dados pessoais usados nos testes devem ser estritamente relevantes para o propósito do teste. Os itens de dados individuais devem ser listados e identificados como:

- Não pessoais
- Pessoais
- Pessoais sensíveis

Uma vez que os dados tenham sido classificados, as razões para inclusão no teste devem ser fornecidas. Quando não for possível encontrar um motivo para a inclusão, os dados pessoais não devem ser usados.

O método de teste precisa garantir que a integridade dos dados seja mantida e quaisquer riscos à precisão sejam mitigados.

É importante que um regime de teste do sistema seja implementado para manter uma trilha de auditoria para capturar erros durante o processo de teste e permitir a correção imediata. As verificações de integridade devem ser realizadas nos dados que estão sendo alimentados no sistema, especialmente se houver a possibilidade de esses dados serem mesclados com outros dados ou realimentados no sistema de origem.

Deve-se ter mecanismos para garantir que os dados de teste não sejam retidos por mais tempo do que o necessário para atender aos requisitos legais, regulamentares ou comerciais. Eles serão destruídos após os testes. Os dados pessoais não serão divulgados para fins de teste a terceiros. Caso isso seja necessário, será solicitada autorização para os titulares dos dados.

10.1. TÉCNICAS UTILIZADAS

Os testes deverão ser realizados utilizando:

- Embaralhamento de chaves primárias;
- Atribuição de valores nulos onde as informações são mais sensíveis;
- Truncar dados;
- Embaralhamento de letras e geração de dados randômicos;
- Nenhuma imagem de produção é usada em ambiente de teste;
- Acesso restrito à equipe de desenvolvimento, mediante autenticação de dois fatores.

10.2. MASCARAMENTO/ANONIMIZAÇÃO DE DADOS

Antes de realizar os testes, devem ser feitas considerações para identificar se dados fictícios ou anonimizados podem ser usados. Se os dados de teste fictícios não puderem ser usados, existem várias técnicas que podem ser utilizadas para mascarar efetivamente os dados para que os riscos sejam mitigados. Algumas técnicas de teste e mascaramento de dados adotadas pela DMS LOGISTICS são elencadas a seguir.

10.2.1. SUBSTITUIÇÃO

Essa técnica consiste em substituir aleatoriamente o conteúdo de uma célula ou coluna de dados por informações que parecem semelhantes, mas não estão relacionadas à informação real. Por exemplo, nomes reais são substituídos por aqueles extraídos de uma lista aleatória.

10.2.2. EMBARALHAMENTO

O embaralhamento é semelhante à substituição, mas substitui-se o valor de uma célula por dados derivados de outras células dessa coluna. Esse método não é tão útil em bancos de dados menores, onde a falta de variedade entre os valores pode causar um problema.

10.2.3. VARIAÇÃO DE NÚMERO E DATA

Este método é útil para valores numéricos e de data. É implementado um algoritmo que modifica cada valor (como data de nascimento) por uma porcentagem aleatória. Essa técnica tem a vantagem de ser uma maneira razoável de mascarar dados, mantendo sua utilidade.

10.2.4. PSEUDONIMIZAÇÃO

Poderá ser usada a pseudonimização, ou “o tratamento de dados pessoais de forma que os dados pessoais já não possam ser atribuídos a um titular de dados específico (a pessoa a quem os dados dizem respeito) sem a utilização de informações adicionais, desde que essas informações adicionais sejam mantidas separadamente e sujeitas a medidas técnicas e organizativas para assegurar que os dados pessoais não são atribuídos a uma pessoa singular identificada ou identificável”.

10.3. ACCOUNTABILITY

A DMS LOGISTICS mantém documentação relativa ao processamento dos dados pessoais para teste, identificando como e se foram cumpridos os princípios da LGPD, quais riscos existem para os dados e como a empresa mitigou esses riscos.

A conformidade é demonstrada das seguintes maneiras:

- Mantendo informação do ativo que está sendo testado;
- Retenção de logs de auditoria e acesso de quando os testes ocorreram.

10.4. PLANO DE TESTES

Para garantir um nível aceitável de segurança nos sistemas da DMS LOGISTICS, deverão ser feitos os seguintes testes:

- Testes utilizando ferramentas automatizadas no ambiente de desenvolvimento sempre no fim de cada sprint.
- Testes de segurança anual feito por um humano para buscar vulnerabilidades que não possam ser detectadas por ferramentas automatizadas (ambiente de produção).

No ambiente de produção devem ser realizados dois tipos de testes:

- Black box
- Gray box

11. ACESSO AO CÓDIGO FONTE E AOS REPOSITÓRIOS

O acesso ao código fonte é restrito, com base no princípio do menor privilégio. O controle de versionamento utilizado pela DMS LOGISTICS é o Git (Atlassian Bitbucket). Todos que lidam com código fonte ou gerenciam/mantêm repositórios de código-fonte devem seguir os seguintes requisitos destinados a proteger o código-fonte contra acesso não autorizado, roubo, alteração e perda.

O código fonte da DMS LOGISTICS nunca deve ser transferido de forma não criptografada.

As senhas e outras credenciais usadas para acessar os repositórios de código fonte são armazenadas em formato criptografado pela gestão de identidades na plataforma Atlassian Bitbucket.

É aceitável armazenar tokens, cookies, tickets ou outras credenciais de autenticação localmente, desde que não contenham senhas não criptografadas (ou equivalentes).

O acesso aos repositórios é feito por meio de criptografia de chave pública, do tipo SSL (Secure Sockets Layer), individual a cada usuário em complemento ao login e senha da plataforma Atlassian Bitbucket. Para obter acesso, se faz necessário solicitá-lo ao administrador dos repositórios.

12. NÃO CONFORMIDADE

Nos casos em que for determinado que ocorreu uma violação das políticas da DMS LOGISTICS, serão tomadas medidas corretivas, incluindo restrição de acesso aos serviços ou instauração de ação disciplinar, o que pode acarretar na demissão.

13. DECLARAÇÃO DE COMPROMETIMENTO

Quando houver novos Dirigentes, Colaboradores, Prestadores de Serviços e Parceiros da DMS LOGISTICS, estes se comprometem a seguir e pôr em prática as Políticas da DMS LOGISTICS.

14. ATUALIZAÇÃO

A Política de Mesa e Tela Limpa da DMS LOGISTICS deve ser atualizada sempre que necessário ou em um intervalo não superior a 01 (um) ano.

HISTÓRICO DE REVISÃO

Revisão	Data	Descrição
00	13/02/2023	Emissão do documento.
01	10/03/2023	Revisão e padronização do documento

APROVAÇÃO E CLASSIFICAÇÃO DA INFORMAÇÃO

Elaborado por:	CyberSecurity Team	
Revisado por:	Leonardo Sabbadim	
Aprovador por:	Victor Gonzaga	
Nível de Confidencialidade:	**X**	**Informação Pública**
		Informação Interna
		Informação Confidencial
		Informação Sigilosa

NUNCA COLOCAMOS EM RISCO A QUALIDADE E NEM A ÉTICA NOS NEGÓCIOS

WE NEVER COMPROMISE ON QUALITY AND BUSINESS ETHICS

WWW.DMSLOG.COM