Norma de Segurança da Informação

Desenvolvimento e Manutenção de softwares e aplicações

Código	**NSI-12**
Versão	V2023.1
Data da Versão	14/08/2023
Criado por	José Jardel Luciano Ferreira da Silva
Aprovado por	Rodrigo Vieira
Nível de confidencialidade	Interno
Número do Controle	

Histórico de alterações

Data Alteração	Versão	Alterado por	Descrição alteração
09/11/2020	V2020.2	José Jardel L. F da Silva	Revisão
12/11/2020	V2020.3	José Jardel L. F. da Silva	Capítulo 11
15/03/2022	V2022.1	André Iesca Rodrigues	Revisão
14/08/2023	V2023,1	Danilo Goes	Revisão
18/12/2024	V2024,1	Danilo Goes	Revisão

Sumário

Norma de Segurança da Informação 1
Desenvolvimento e Manutenção de softwares e aplicações 1
Histórico de alterações 1
1 Introdução 3
2 Diretrizes 3
3 Definições 3
4 Security Readiness Policy 3
5 Security Requirements Policy 4
5.1 Práticas 4
5.1.1 Define Security Requirements 4
5.1.2 Security Backlog 4
5.1.3 Evil User Stories 4
5.1.4 Protection Poker 5
5.1.5 Perform Risk Analysis 5
5.1.6 Security Coding Rules 5
5.1.7 Security Code Review 5
5.1.8 Pair Programming 6
5.1.9 Security Automated Testing 6
5.1.10 Security Code Analysis 6
6 Disposições Gerais 7
7 Desenvolvimento Terceirizado 9
8 Testes 10
9 Aceitação de Software 10
9.1 Mudanças Técnicas no Ambiente de Produção 10
10 Implantação 11
11 Proteções 11
12 Revisões 12
13 Gestão da Norma 12

1 INTRODUÇÃO

A Norma de segurança da informação **NSI-12** complementa Política Geral de Segurança da Informação, definindo as diretrizes para desenvolvimento e manutenção de softwares e aplicações no ambiente da YUP CHAT, incluindo, caso existam, desenvolvimento de equipes terceirizadas.

2 DIRETRIZES

Estabelecer as diretrizes que regulamentam a segurança para o processo de desenvolvimento e manutenção de software no âmbito da YUP CHAT.

3 DEFINIÇÕES

3.1 **Artefato de Software:** item criado como parte da definição, manutenção ou utilização de um processo de software, incluindo, entre outros, descrições de processos, planos, procedimentos, especificações, projetos de arquitetura, projeto detalhado, código, documentação para o usuário.

3.2 **Base de Dados**: conjunto de dados organizados de forma a servir de base para que o usuário processe e recupere informações.

3.3 **Gestão de Configuração:** conjunto de procedimentos técnicos e gerenciais que são definidos para identificação de Ativos de Tecnologia da Informação e para a gestão de suas alterações.

3.4 **Rastreabilidade:** capacidade de acompanhamento e registro de todos os eventos e movimentações ocorridas, desde a criação da informação até o seu descarte. Usuário: qualquer colaborador, seja ele servidor, estagiário, cliente, parceiro, fornecedor, prestador de serviço ou terceiro em geral que acessa informações ou utiliza recursos de Tecnologia da Informação disponibilizados pela YUP CHAT.

4 SECURITY READINESS POLICY

4.1. Ter um time de desenvolvimento, apto a lidar com aspectos de segurança de software, é primordial para entregar um produto que atenda aos requisitos dessa área, principalmente, em se tratando de times ágeis.

4.2. Essa política propõe as principais práticas do processo de desenvolvimento de software, com o intuito de conscientizar e orientar times ágeis, acerca das características relevantes no concernente a segurança da informação.

4.3. Apesar do enfoque no time de desenvolvimento, se possível, essa política deve ser estendida, a todos os envolvidos no processo de desenvolvimento (Clientes, Stakeholders e outros).

5 SECURITY REQUIREMENTS POLICY

- Com o intuito de mitigar problemas relacionados a falhas de segurança, causados pela falta de evidenciação desses requisitos, essa política propõe práticas, que incluam requisitos de segurança como parte do processo de times ágeis. Esta tarefa, é essencial desde o início do projeto, sendo perpetuada durante as iterações até a conclusão do software

5.1 Práticas

- Esta prática visa identificar funcionalidades relacionadas à segurança, expressas de forma explícita, dentro de um projeto de software . É necessário garantir que requisitos de segurança sejam discutidos pelo time de desenvolvimento, junto ao cliente e a outros stakeholders.

5.1.1 Define Security Requirements

- Esta prática visa identificar funcionalidades relacionadas a segurança, expressas de forma explícita, dentro de um projeto de software. É necessário garantir que requisitos de segurança sejam discutidos pelo time de desenvolvimento, junto ao cliente e a outros stakeholders.

5.1.2 Security Backlog

- Após o levantamento dos requisitos de segurança do projeto, pode ser criado um Security Backlog, que deve conter a lista de atividades relativas aos itens de segurança apontados

5.1.3 Evil User Stories

- Evil User Stories descrevem cenários de ameaças de segurança ao sistema, demonstrando como o sistema pode ser afetado por atacantes. O objetivo principal é estressar as atividades do Product Backlog e do

Security Backlog, de modo a detectar possíveis vulnerabilidades e fraquezas dos sistemas que podem ser exploradas por usuários mal-intencionados.

5.1.4 Protection Poker

- Essa prática baseada no Planning Poker tem o intuito de priorizar atividades de desenvolvimento de acordo com o risco de segurança a elas associado. O valor dado a cada atividade deve seguir a sequência de fibonacci. No Protection Poker quanto maior o valor dado, maior o risco de segurança associado.

5.1.5 Perform Risk Analysis

- Analistas de segurança analisam o risco de segurança para os requisitos do projeto, de modo que seja possível classificá-los de acordo com o nível de risco apresentado. Desconsiderar a análise de risco nas fases iniciais leva a problemas caros no futuro. O risco, normalmente, deve ser determinado de aplicação para aplicação. O objetivo final, seria realizar o levantamento de todos os riscos

5.1.6 Security Coding Rules

- Diversos times de desenvolvimento aplicam práticas de implementação como forma de manter a qualidade do código, sendo que, uma das mais difundidas é o Clean Code; de modo semelhante essas regras podem ser definidas em relação segurança. Durante a fase de implementação dos requisitos do software muitas vulnerabilidades podem ser inseridas de modo não intencional.

5.1.7 Security Code Review

- Times de desenvolvimento, muitas vezes, usam Code Review como prática para estabelecer e manter padrões de qualidade de código. Está prática consiste na realização de uma revisão manual do código de

desenvolvimento, sendo esta executada por outros membros do time, antes que esse código vá para os repositórios.

5.1.8 Pair Programming

- Pair Programming é um conceito ágil que define a situação onde os desenvolvedores codificam em pares, resolvendo e revisando problemas de forma conjunta, enquanto o código é escrito. Desenvolver dessa forma leva os desenvolvedores a compartilharem conhecimentos.

5.1.9 Security Automated Testing

- Testes automatizados são uma prática bastante implantada em diversos projetos de software. O intuito dessas práticas é implementar funções de verificação que comprovam a corretude dos requisitos implementados. Aproveitar a implementação desses testes e incluir validação de funcionalidades relativas à segurança pode aumentar a confiabilidade do software. Além da validação do código implementado, outro benefício do software é adicionar uma certeza racional, que quando mudanças ocorrerem, o teste deverá garantir que o código, já validado, continue funcionando como deveria

5.1.10 Security Code Analysis

- O código fonte é um artefato produzido por todos os projetos de software. Garantir que esse código seja analisado em busca de falhas de segurança é essencial para validar a segurança do software implementado. Essa prática consiste, basicamente, em três metodologias de verificação, quais sejam: Static Code, Analysis, Dynamic Code Analysis e Fuzzy Testing.

- A primeira analisa o código puro, a partir de regras de verificação predefinidas, em busca de trechos de código que apresentem problemas de segurança apontados pelas regras adotadas. A Segunda analisa o código, em tempo de execução, juntamente como todo o código de bibliotecas e de terceiros incorporados ao código implementado.

6 DISPOSIÇÕES GERAIS

Pelo menos 1 (uma) metodologia deve ser estabelecida para todo desenvolvimento ou manutenção, com base nas melhores práticas de mercado, contemplando, entre outros:

- Planejamento;
- Análise de requisito;
- Projeto;
- Codificação;
- Revisão;
- Compilação;
- Teste.

6.1 Todo desenvolvimento ou manutenção de software deve ser formalmente autorizado.

6.2 Todo desenvolvimento de sistema para aplicativos mobile, deve-se ofuscar o código (Ex: dexguard), sslcheck (SSL Pinning) e tamper-proof

6.3 Para todo desenvolvimento ou manutenção de software deve ser realizada uma análise de impacto.

6.4 Toda alteração de escopo de desenvolvimento ou manutenção de software deve ser documentada e formalmente autorizada.

6.5 Todas as ferramentas de desenvolvimento devem ser homologadas e licenciadas.

6.6 Todo projeto de software deve conter um documento de especificação que descreva seus requisitos de segurança, os quais devem, entre outros, contemplar:

a. Mecanismo de autenticação do usuário, que deve utilizar senhas com métrica mínima e exigir do usuário a troca periódica da senha;

b. O mecanismo de autenticação do usuário, que deve bloquear o acesso após número definido de tentativas de login/autenticação com falha;

c. A verificação da senha por meio de mecanismo que impeça fraudes de repetição, interceptação ou quebra de integridade na comunicação entre o cliente e o servidor;

d. A escolha da senha por novos usuários sem a interferência do pessoal de apoio ou o recebimento pelos mesmos, de uma senha inicial que precise ser trocada;

e. O armazenamento da senha pelo sistema, de forma criptografada e irreversível;

f. A uniformidade do controle de acesso em todo o sistema, utilizando-se uma única rotina de verificação;

g. A realização do controle de acesso na camada mais próxima possível dos dados;

h. O registro, pelo sistema, dos eventos significativos para a segurança, principalmente, início e fim do mecanismo de auditoria;

i. O registro, pelo sistema, das falhas de login, indicando o número de tentativas;

j. O registro, pelo sistema, da criação e remoção de usuários, bem como da atribuição e da remoção de direitos do usuário;

k. A proteção da trilha de auditoria contra remoção e alteração por parte de todos os usuários, exceto dos administradores de auditoria;

l. A capacidade de tolerância do sistema à falhas e retorno a operação;

m. A inexistência, em aplicações web, de dados sensíveis em campos ocultos ou cookies;

n. A realização das verificações e validações de segurança no servidor, em aplicações web;

o. O acesso aos desenvolvedores apenas aos códigos fontes necessários para a alteração, quando autorizados pelo superior imediato;

p. A maior semelhança possível do ambiente de homologação ao ambiente de produção;

q. A exigência de que os aplicativos só passem do desenvolvimento para a homologação após verificação da existência e adequação de sua documentação;

r. A existência de documentação de instalação, configuração e operação do sistema, ressaltando os aspectos de segurança, que deve ser mantida atualizada. Requisitos funcionais, não funcionais e de domínio devem ser especificados e documentados, bem como as manutenções necessárias, considerando os requisitos de segurança definidos no desenvolvimento do sistema.

6.7 A especificação dos requisitos deve ser elaborada em conjunto com a área de negócio solicitante da demanda.

6.8 Um mecanismo de controle de versão deve ser implementado durante o processo de desenvolvimento e manutenção de software.

6.9 Deve existir um programa de conscientização em Segurança da Informação para todos os usuários envolvidos nos processos de desenvolvimento de aplicações.

6.10 Devem existir mecanismos de verificação de vulnerabilidades no código fonte durante o processo de desenvolvimento e manutenção de software.

6.11 Incidentes de segurança devem ser abertos quando vulnerabilidades forem identificadas durante o processo de desenvolvimento e manutenção de software.

6.12 O acesso aos códigos fontes deve ser controlado e restrito aos desenvolvedores envolvidos, em seus respectivos projetos.

6.13 Os códigos fontes não devem conter identificações e/ou senhas de acesso às bases de dados, sejam elas de teste, de homologação ou de produção.

6.14 Ambientes de desenvolvimento e testes, de homologação e de produção devem ser isolados entre si.

6.15 Um processo de gestão de configuração deve ser implementado e deve abranger todo o processo de desenvolvimento e manutenção.

6.16 Todo software que implique em manipulação de dados deve ser desenvolvido com controle de acesso lógico. Mecanismos adicionais que possibilitem a rastreabilidade das operações efetuadas devem ser considerados em casos de manipulação de dados sensíveis.

6.17 Não deve incluir credenciais, usuários e senhas, chaves de API, parâmetros de negócio, hard coded armazenadas dentro ou mapeadas para uma página web, script, chave de função ou qualquer outro tipo de arquivo de código fonte.

7 DESENVOLVIMENTO TERCEIRIZADO

7.1 Todos os contratos com terceiros devem contemplar cláusulas de sigilo e confidencialidade.

7.2 Os produtos desenvolvidos externamente devem obedecer a padrões e metodologias homologadas, além de atender aos requisitos funcionais, não funcionais, de domínio e de segurança definidos.

7.3 O contrato de desenvolvimento de produtos com terceiros deve prever, no mínimo, os artefatos de software a serem entregues em cada fase, validação, o procedimento de aceite final e o período de garantia.

8 TESTES

Procedimentos de testes no software devem ser definidos e utilizados par todo desenvolvimento ou manutenção realizados, e devem contemplar, entre outros, controles tais como:

A. Validação de dados de entrada;

B. Controle de processamento interno;

C. Integridade de mensagens;

D. Validação de dados de saída

8.1 Os testes devem validar os mecanismos de segurança especificados no desenvolvimento ou na manutenção do software.

8.2 Os testes de aceitação do software devem ser realizados por uma equipe diferente da equipe desenvolvedora, que deve ser composta por usuários da área de desenvolvimento e da área de negócio solicitante.

8.3 A utilização de dados de produção em ambiente de testes deve ser autorizada formalmente.

8.4 As informações contidas na base de dados de ambiente de produção, se utilizadas para testes, devem sofrer alterações, de modo a preservar sua confidencialidade.

9 ACEITAÇÃO DE SOFTWARE

Os artefatos de software, provenientes de desenvolvimento ou manutenção, devem ser homologados antes de serem utilizados em ambiente de produção.

9.1 Mudanças Técnicas no Ambiente de Produção

As atualizações de configuração no ambiente de produção devem ser realizadas, inicialmente, em ambiente de teste e, todo software deve ser analisado criticamente, considerando os seguintes aspectos:

- Análise crítica dos procedimentos de controle e integridade do software, garantindo que os mesmos não foram comprometidos pelas mudanças efetuadas no ambiente de produção;

- Revisão do planejamento e do orçamento anual para suporte, garantindo investimentos para revisões e testes de softwares resultantes das modificações do ambiente de produção;

- Revisão do Plano de Continuidade dos Negócios para contemplar mudanças necessárias resultantes das modificações do ambiente de produção.

10 IMPLANTAÇÃO

10.1 Os produtos homologados devem ser implantados em ambiente de produção, por meio de procedimentos técnicos definidos pela área de Tecnologia da Informação do órgão ou entidade e aceite da área cliente.

10.2 Planos de Continuidade Operacional devem ser desenvolvidos pelas áreas de negócio do órgão ou entidade para garantir a continuidade dos processos envolvidos nas implantações de software ou outras mudanças relacionadas.

10.3 A implantação de novo software deve ser realizada de acordo com o calendário definido pelas áreas de negócio do órgão ou entidade, com a participação da respectiva área de Tecnologia da Informação.

11 PROTEÇÕES

11.1 A equipe de desenvolvimento deve utilizar de meios de proteção, por exemplo ofuscação/mascaramento do código fonte, assim, não utilizando os dados de produção em outro ambiente de forma legível.

11.2 Ao transferir o código para o setor de **"Teste" ou "Homologação",** o código fonte deve ser ofuscado, como citado anteriormente

11.3 Fica **PROIBIDO** o uso ou transferência de informações (códigos fontes e documentações sem ofuscação/mascaramento) fora do ambiente de produção.

12 REVISÕES

Esta norma é revisada com periodicidade anual ou conforme o entendimento do Comitê Gestor de Segurança da Informação.

13 GESTÃO DA NORMA

- A norma **NSI-12** é aprovada pelo Comitê Gestor de Segurança da Informação, em conjunto com a Diretoria da YUP CHAT.
- A Assim como a ética, a segurança deve ser entendida como parte fundamental da cultura interna da YUP CHAT. Ou seja, qualquer incidente de segurança subtende-se como alguém agindo contra a ética e os bons costumes regidos pela instituição.
- A presente norma foi aprovada no dia 02/11/2020.