Guia para Desenvolvimento Seguro de Software

Tribunal Regional do Trabalho da 4a Região

Versão	Data	Autor	Descrição
1.0	01/09/17	Alexandre Leite, Anderson Miranda, André Farias, Felipe Giotto, Felipe Levin, Michel Barreto e Stéfano Mór.	Versão inicial.
1.0.1	30/11/17	Stéfano Mór	Inclusão de verificação periódicas de listas de vulnerabilidades no *roadmap*.
1.0.2	24/04/18	Stéfano Mór	Unificação das seções “Determinação de Identidade e Nível de Acesso de Usuários” com “Autenticação de Usuários”.

O presente documento deve ser revisado ao menos anualmente.

APRESENTAÇÃO E OBJETIVOS

Este documento é o guia para desenvolvimento seguro de *software* no âmbito do Tribunal Regional do Trabalho da 4a Região (TRT4). Seu objetivo é servir como guia de boas práticas a serem adotadas por analistas, desenvolvedores e instaladores de *software,* tornando o processo de concepção dos sistemas construídos dentro deste Tribunal mais confiável, auditável, estável e protegido contra ameaças. As orientações aqui contidas são direcionadas a todos os envolvidos no processo de desenvolvimento de *software* no âmbito do TRT4.

As diretrizes constantes no presente instrumento foram elaboradas por um Grupo de Trabalho para Desenvolvimento Seguro de *Software* (gtDevSeg), subordinado diretamente à Coordenadoria de Desenvolvimento de Sistemas (CDS) da Secretaria de Tecnologia da Informação e Comunicações (SETIC) do TRT4.

O presente guia funciona como complemento em relação às normas em vigor, sobretudo no que se refere à Portaria No 4.772 de 23 de setembro de 2008 do TRT4 e à norma ABNT NBR ISO-IEC 27002:2013.

FORMATO

O documento é estruturado em torno de “diretrizes”, recomendações de boas práticas a serem seguidas em cada um dos tópicos listados. Seu formato é detalhado abaixo. Após, discute-se a estrutura do documento e disposição de seu conteúdo.

Diretrizes

Uma diretriz é escrita em forma imperativa ou imperativa negativa --- “*Deve-se* [...]” ou “*Não se deve* [...]” e pode estar classificada, em função da necessidade de proteção aos dados e das obrigações imputadas ao programador ou analista, em três níveis --- cada um *cumulativo* com as medidas do nível anterior, salvo em sobreposição de escopo. Os níveis são:

- **Mínimo**. São *deveres* a serem seguidos na construção de sistemas para que se obtenha um nível de segurança considerado empiricamente mínimo.

- **Padrão**. São *recomendações* pertinentes à construção de sistemas para que se obtenha um nível de segurança considerado empiricamente padrão à data da última revisão do documento.

- **Forte**. São *medidas adicionais* pertinentes à construção de sistemas para que se obtenha um nível de segurança empiricamente forte à data da última revisão do documento.

Na ausência da indicação de níveis de segurança em uma dada diretriz seu nível é considerado *mínimo*.

Estrutura

Este guia é direcionado, sobretudo, à desenvolvedores de *software* inseridos na construção dos sistemas do TRT4. Considerando esse cenário, o guia foi projetado para facilitar a consulta expressa, mas sem omitir informações detalhadas. O documento começa

com um guia de referência rápida --- *que somente traz as orientações mais relevantes* ---, seguido das diretrizes principais. Um glossário está disposto ao final do documento e deve ser utilizado em caso de necessidade de detalhamento das seções iniciais.

ATUALIZAÇÕES

O resultado de cada revisão do presente instrumento busca acompanhar, sobretudo, tópicos relevantes para a realidade do desenvolvimento de sistemas do TRT4 e não se furta em revisar seus requisitos conforme esta realidade se transforma. O mecanismo que emprega para isto é a revisão constante de seus conteúdos.

Ao final deste guia há um *roadmap* para atualizações. Ele contém tópicos e ponderações a serem desenvolvidos em futuras versões do documento, *a priori*. Ele serve de base para as atualizações e deve, portanto, ser também atualizado a cada nova versão do documento. Neste *roadmap* constam tópicos não abordados e detalhamentos de tópicos já discutidos, bem como considerações sobre o formato do documento a serem ponderadas nas próximas atualizações.

A seguir estão listadas linhas gerais que devem ser observadas quando da atualização do documento:

- Deve-se descrever as diretrizes de maneira breve. Detalhes podem ser incluídos nos anexos técnicos ao final do documento.
- Deve-se utilizar a estrutura de tópicos do formato do documento (seus estilos; Título, Legenda, Corpo, *etc*.) sempre que possível.
- Deve-se escrever diretrizes na forma negativa sempre que possível.
- Deve-se reportar à chefia qualquer discordância das práticas aqui descritas com portarias e/ou leis cujo conteúdo e normatização se sobreponha às diretrizes aqui apresentadas ou no caso destas adentrarem a competência de outros setores do TRT4.
- Deve-se adicionar conteúdo partindo das medidas que provém o nível de segurança mínimo referente àquele aspecto; níveis mais elevados de segurança (padrão, forte) devem ser descritos depois ou elencados no *roadmap*, se necessário.
- Deve-se escrever diretrizes objetivas, um ponto de cada vez, mas com a aplicabilidade mais geral possível.
- Deve-se incluir a motivação e explicações preliminares sobre uma diretriz, se necessário, no preâmbulo de sua seção.

GUIA REFERÊNCIA RÁPIDA

Tópico	Descrição	Diretriz **Mínimo**	Diretriz **Padrão**	Diretriz **Forte**
Armazenamento de Dados	Armazenamento para Dados Abertos	Acesso de escrita restringido por senha.	-	-
Armazenamento de Dados	Armazenamento para Dados Fechados	Acesso de leitura/escrita restrito por senha.	-	Deve-se armazenar dados criptografados.
Armazenamento de Dados	Permissões de Acesso a Dados em Banco	Aplicação não deve utilizar usuário root.	Aplicação não deve ter permissões DDL, somente permissões estritamente necessárias.	Correspondência 1-1 entre usuário de sistema e de banco.
Armazenamento de Dados	Gerenciamento e Distribuição de Senhas para Acesso a Dados	Senhas devem seguir padrão do TRT; não devem ser armazenadas em código fonte	Senhas devem seguir padrão deste documento; Não utilizar mesma senha para homologação e produção; Salvar de forma segura dados de usuários e sistemas que utilizam a senha	
Controle de Usuários: Acessos e Permissões	Identidade do Usuário e Nível de Acesso	Usuário e senha nominais.	Dar ciência das permissões e níveis de acesso. Utilizar grupos do AD.	Utilizar certificado digital.
Controle de Usuários:	Autenticação de Usuários	Não armazenar senhas em texto	Deve-se utilizar autenticação via	

Acessos e Permissões		plano sem utilizar um algoritmo de hash seguro e salt.	AD e/ou o framework OAuth2 sempre que possível para autenticar usuários internos.	
Controle de Usuários: Acessos e Permissões	Autenticação em Sistemas Web	HTTPS pelo menos nas telas de login	HTTPS em todo o sistema e verificações adicionais.	
Comunicação Segura	Comunicação entre sistemas e/ou módulos	controle de duplicação e integridade da informação	controle de autenticação e confidencialidade	controle para não-repúdio e registro de entrega
Ataques à Sistemas e suas Defesas	Prevenção de ataques	Prevenir SQL Injection, HTML Injection e Javascript Injection	Prevenir ataques XSS, de quebra de autenticação e gerenciamento de sessão	Submeter sistema a ferramentas de testes de invasão
Auditoria, Rastreamento e Logs	Rastreamento das operações realizadas pelos usuários nos sistemas	Documento de requisitos do *software* deverá definir as informações a serem armazenadas e o local de armazenamento.	Documento de requisitos do *software* deverá definir políticas de retenção e revisão dos logs.	
Prevenção, Reação e Mitigação de Falhas de Segurança	Diretivas de backup	Incluir no plano de projeto as necessidades e responsabilidades de backup de dados e código-fonte	Definir procedimento e capacitar responsáveis pela restauração de backups	Criar baselines de versões e realizar simulações de restauração de dados continuamente
Prevenção, Reação e Mitigação de Falhas de Segurança	Políticas de testes	Realizar testes manuais antes de liberações de versão de software	Elaborar testes automatizados, cenários de testes e outras políticas que garantam segurança, sigilo e não vulnerabilidade do software	Propor constantes desafios com intuito de identificar falhas de segurança nos softwares
Prevenção, Reação e Mitigação de Falhas de Segurança	Ocorrências de falhas de segurança	Definir política para imediata indisponibilização do sistema e correção da falha	Política de acompanhamento pós-ocorrência	Revisão contínua da política de testes com base em lições aprendidas

Criptografia e Hash	Tamanho de chave para cifradores simétricos/assimétricos.	128/1024 bits	192/2048 bits	256/4096 bits
Criptografia e Hash	Modo de cifrador de bloco.	Mais seguro que ECB	-	-
Criptografia e Hash	Requisitos cifradores.	Somente chave sigilosa, melhor ataque força-bruta.	Não usar cifradores/modos obsoletos. *E.g.*, DES, RC4, *etc.*	-
Criptografia e Hash	Requisitos função de *hash* criptográfico.	Usar *salt* sempre que possível.	Não usar cifradores/modos obsoletos. *E.g.*, MD5, SHA1, *etc.*	-
Senhas	Tamanho de senhas.	8 caracteres	12 caracteres	20 caracteres
Senhas	Variação de tipos de caracteres: letras maiúsculas, letras minúsculas, dígitos, símbolos	2 dos 4 tipos ao menos	Letras maiúsculas e minúsculas mais um 1 dos 2 tipos restantes ao menos.	Mistura de todos os tipos.
Senhas	Geração.	Não utilizar senhas comuns. *E.g.*, 12345, datas de aniversário.	Usar *software* gerador de senhas.	Usar *software* validador de senhas diferente do gerador.
Senhas	Periodicidade de troca.	Não superior a 1 ano.	Não superior a 6 meses.	-
Senhas	Armazenamento	Senhas devem ser armazenadas criptografadas e com hash.	Criptografia usada para o armazenamento com a descrita no nível *padrão*.	Criptografia usada para o armazenamento com a descrita no nível *forte*.
Senhas	Número permitido de tentativas de validação.	Não superior a 5 tentativas por minuto.	Senha bloqueada em caso de 5 erros de validação consecutivos.	Exigir prova de de origem da requisição (*e.g.*, *captcha*, assinatura digital) após a primeira falha.
Ciclo de Vida de *Software*	Projeto e desenvolvimento.	Deve haver etapa de modelagem de riscos de segurança, com	-	-

		verificações periódicas no cronograma.		
Ciclo de Vida de *Software*	Documentação e codificação.	Documentar medidas de segurança, inclusive no código da aplicação.	-	-
Comunicação inter-sistemas	Comunicação segura entre sistemas e módulos	HTTPS	HTTPS, certificado digital, banco de dados, VPN	WS-ReliableMessaging
Ambiente de desenvolvimento	Armazenamento do código fonte	Sistema de controle de versão	Sistema de controle de versão distribuído	
Ambiente de desenvolvimento	Acesso ao código fonte	Servidores da SETIC	Definir com chefia caso-a-caso.	
Ambiente de desenvolvimento	Segregação dos ambientes (DEV,PRD,HOM)	Banco de dados e servidor de aplicação individualizados	-	Acesso restrito ao ambiente de produção
Ambiente de desenvolvimento	E-mails dos sistemas	E-mail criado especificamente para o sistema	-	-

DIRETRIZES

ARMAZENAMENTO DE DADOS

Esta seção apresenta definições e diretrizes que tratam do armazenamento de informações sigilosas ou não e de sua disponibilização. Define taxonomia para classificação de dados e descreve procedimentos para o armazenamento seguro dessa informação em *bancos de dados*. Detalha o gerenciamento de permissões de acesso e distribuição de senhas a serem adotadas para operacionalização essas estruturas.

Sigilo

No escopo deste documento os dados serão classificados, quanto ao seu *sigilo*, como:

- **Abertos.** Dados públicos (informação pública, conforme Portaria 4.772/2008 do TRT4), cujo conteúdo pode ou deve ser divulgado ao público externo.

- **Fechados.** Dados cujo acesso é restrito a um grupo específico de pessoas (informação secreta, sigilosa ou interna, conforme Portaria 4.772/2008 do TRT4).

Procedimentos e Meios para Armazenamento de Dados

Diretrizes para armazenamento de dados *abertos*:

Mínimo

- Não se deve utilizar meio de armazenamento que não possua acesso para escrita restrito por senha.

Diretrizes para armazenamento de dados *fechados*:

Mínimo

- Não se deve utilizar meio de armazenamento que não possua acesso para leitura e escrita restrito por senha.

Forte

- Deve-se armazenar dados criptografados.

Permissões para Acesso a Informações em Bancos de Dados

Diretrizes que tratam de permissões de acessos às informações contidas em bancos de dados por usuários e aplicações.

Observação. Deve ser dada especial atenção às permissões de acesso das tabelas de auditoria do sistema.

Observação. A alocação de permissões para usuários da aplicação deve ser feita em função dos recursos disponíveis (*eg*, quando existe somente um usuário de banco de dados disponível para acessar o sistema), *mensurados ainda na fase de projeto de software.*

Mínimo

- Não se deve disponibilizar à aplicações acesso à algum banco de dados utilizando *login* de usuário com permissões de *root*.

Padrão

- Não se deve disponibilizar à aplicações acesso a algum banco de dados utilizando *login* de usuário com permissões para execução de comandos em *Data Definition Language* (DDL).

- Não se deve disponibilizar à aplicações acesso à algum banco de dados utilizando *login* de usuário com permissões além das estritamente necessárias ao seu funcionamento.

Forte

- Deve haver correspondência um-para-um entre cada usuário de uma dada aplicação e do banco de dados.

Gerenciamento e Distribuição de Senhas para Acesso a Dados

Diretrizes para o gerenciamento e uso das senhas que protegem os dados armazenados:

Mínimo

- Não se deve permitir a elaboração de senhas que não sigam os padrões estabelecidos na Portaria 4.772/2008 do TRT4.

- Não se deve utilizar o armazenamento de senhas em código-fonte.

Padrão

- Não se deve permitir a elaboração de senhas que não sigam os padrões estabelecidos neste documento.

- Deve-se armazenar de forma segura os dados de usuários e os sistemas que utilizam cada senha fornecida.

- Não se deve utilizar as mesmas senhas para ambientes de desenvolvimento ou homologação e produção.

GERENCIAMENTO DE ACESSOS E PERMISSÕES DE USUÁRIOS

Esta seção apresenta definições e diretrizes que tratam do controle de acesso aos dados impostos aos usuários e a atribuição das permissões necessárias.

Autorização e Autenticação de Usuários

Diretrizes para a verificação da identidade de usuários ao realizarem operações nos sistemas e para determinação de identidade do usuário e seu correspondente nível de acesso às informações.

Mínimo

- Não se deve armazenar senhas em texto plano sem utilizar um algoritmo de *hash* seguro e *salt*.
- Deve-se utilizar controle de usuário e senha nominais para determinar a identidade do usuário.

Padrão

- Deve-se utilizar autenticação via AD e/ou o *framework* OAuth2 sempre que possível para autenticar usuários internos.
- Deve-se dar ciência ao usuário das permissões e níveis de acesso que possui.
- Deve-se utilizar grupos de *Active Directory* (AD) para determinar as políticas de acesso e *roles* de usuário.

Forte

- Deve-se utilizar certificado digital para determinar a identidade do usuário.

Observação. Autenticação AD *versus* OAuth2.

- No OAuth2, ao contrário do AD, a autenticação é feita diretamente em uma página do externa, via HTTPS, sendo que o sistema web *não tem acesso* às credenciais inseridas pelo usuário. Depois do login, a página externa retorna um *token* para a aplicação. Esse *token* garante que o usuário foi autenticado corretamente.
- No OAuth2 é exigida uma conexão ativa com a Internet. Em situações de contingência onde o usuário não tenha acesso à página externa, não será possível autenticar-se no sistema.

Autenticação em Sistemas Web

Diretrizes específicas para autenticação em sistemas *web*, que realizam esse controle fazendo a informação trafegar em meios de acesso compartilhados, utilizando protocolos do nível de Aplicação OSI (*eg*, HTTP).

Observação. Sendo o HTTP um protocolo *stateless*[1], que utiliza *cookies*[2] para manter sessões de usuário, faz-se necessário garantir tanto a segurança da troca de credenciais (*login*) quanto a segurança das demais páginas acessadas pelos usuários dos sistemas web. O protocolo HTTPS visa contribuir para que essa segurança seja garantida. Os

[1] Mais informações em https://pt.wikipedia.org/wiki/Protocolo_sem_estado.
[2] Mais informações sobre *cookies* em https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies.

sistemas web podem utilizar certificados que o TRT4 já possua ou certificados fornecidos por autoridades certificadoras (alguns, inclusive, gratuitos[3]).

Mínimo

- Deve-se utilizar HTTPS para controle de autenticação ao menos nas telas de *login* do sistema.

Padrão

- Deve-se utilizar HTTPS em todas as telas do sistema.

- Não se deve limitar o controle de autenticação em sistemas *web* ao uso de HTTPS, empregando-se outras o acesso dos usuários pode (conforme o sistema) ser restringido, ainda, por outros critérios. *Eg*., garantir que, uma vez autenticado, o usuário não possa acessar o sistema de outro endereço IP, a menos que se autentique novamente.

[3] Há sites como “Let’s Encrypt” (https://letsencrypt.org/) que fornecem certificados gratuitos.

COMUNICAÇÃO SEGURA

Esta seção apresenta definições e diretrizes que tratam da transmissão segura de dados sensíveis entre sistemas, de modo a salvaguardar a integridade, autenticidade e demais atributos pertinentes ao uso dos dados comunicados.

Diretrizes para assegurar a segurança da comunicação de dados em função das características apresentadas, conforme os níveis elencados abaixo, de maneira cumulativa.

Mínimo

- Deve-se empregar canal de comunicação com controle de duplicação e perda de informações/mensagens.

- Deve-se empregar canal de comunicação que provenha controle de integridade dos dados transmitidos (*eg*, HTTPS).

Padrão

- Deve-se empregar canal de comunicação com controle de autenticação (*eg*, HTTPS, certificados digitais gerados por autoridades confiáveis, VPNs).

- Deve-se armazenar de maneira segura os dados a serem transmitidos em ambas as extremidades da comunicação (vide a respectiva seção deste documento).

- Deve-se empregar canal de comunicação que provenha confidencialidade dos dados transmitidos (*eg*, HTTPS, VPNs).

Forte

- Deve-se empregar canal de comunicação que provenha garantia de não-repúdio dos dados transmitidos (*eg*, certificados digitais emitidos por entidade confiáveis).

- Deve-se utilizar *logs* confiáveis das informações transmitidas, com confirmação de entrega e recepção das mensagens (*eg*, *WS-ReliableMessaging para SOAP WS*).

ATAQUES À SISTEMAS E SUAS DEFESAS

Esta seção apresenta diretrizes para reforçar a resiliência de sistema a ataques contra sistemas e aplicações.

Observação. Recomenda-se que sejam prevenidos os principais ataques conhecidos em relação à segurança do sistema, de forma a evitar que ataques mal intencionados possam comprometer a segurança do sistema, expor dados sigilosos e realizar operações não autorizadas, dentre outras vulnerabilidades.

Mínimo

- Deve-se prevenir ataques de injeção de SQL *(SQL Injection)*.
    - Não se deve criar SQLs concatenando parâmetros *textuais* de origem não-segura, como parâmetros preenchidos pelo usuário ou mesmo armazenados no banco de dados.
    - Deve-se restringir permissões de acesso ao banco de dados para o usuário da aplicação.
    - Deve-se, sempre que possível, passar parâmetros em comandos SQL (DML ou DDL) utilizando *prepared statements*. Consultas que não podem ser parametrizadas deverão receber tratamento especial, como *escapes* ou codificação em hexadecimal[4].

- Deve-se prevenir ataques de injeção de HTML e Javascript.

Padrão

- Deve-se prevenir ataques do tipo *cross-site scripting* (XSS)[5].

- Deve-se prevenir ataques de quebra de autenticação e gerenciamento de sessão (*Broken Authentication and Session Management*).

Forte

- Deve-se submeter os sistemas a ferramentas de testes de invasão.

[4] A forma de utilização de *PreparedStatements* varia para cada linguagem. Mais detalhes podem ser encontrados em https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet.
[5] Fonte: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

AUDITORIA, RASTREAMENTO E LOGS

Esta seção apresenta diretrizes para a manutenção de registros/logs para posterior auditoria, rastreamento e consulta de incidentes ligados à segurança dos sistemas. Cada sistema possui uma criticidade diferente no que se refere a restrição de acesso a dados, não-repúdio e histórico de operações realizadas no banco de dados. Por esse motivo, essa seção não *define* quais informações devem ser auditadas, mas sim *sugere* possíveis itens que podem ser auditados, rastreados ou *logados*. Estes itens, então, devem ser avaliados pelos gestores do produto.

Observação. Exemplos de eventos que podem ser registrados:

- operações de login e logout;
- acessos a determinadas telas ou seções do sistema;
- acesso a informações com alguma restrição (*eg*, documentos sigilosos, processos em segredo de justiça, dados pessoais ou bancários);
- operações de inclusão, alteração ou exclusão de registros no banco de dados;
- alteração de perfil de acesso (para sistemas que possuem acesso com diferentes perfis); e
- execução de jobs e tarefas automatizadas.

Observação. Exemplos de informações que podem ser armazenadas, relativas a cada evento:

- data e hora;
- usuário que efetuou a operação;
- endereço IP;
- identificador da sessão do usuário (quando aplicável, *eg*, *cookie*);
- tela (página) do sistema de onde a operação foi realizada;
- identificador da instância (para sistemas *clusterizados*);
- para operações de inserção, alteração ou exclusão, o tipo da operação, nome da tabela que foi manipulada, ID do registro e, se for o caso, valores anterior e atual de cada campo;
- parâmetros informados pelo usuário (*eg*, parâmetros GET ou POST), tomando cuidado de não armazenar dados sensíveis, como senhas;
- tempo de resposta do sistema;
- para execução de jobs e tarefas automatizadas, armazenar o resultado da operação; falha, sucesso, cancelada, etc.

Observação. Exemplos de forma de captura dos dados para auditorias:

- alterações aplicadas no banco de dados podem ser auditadas via *triggers*[6];
- auditar as alterações a partir da própria aplicação[7] --- algumas informações poderão não ser registradas (*eg*, operações SQL realizadas por fora da aplicação).

---

[6] Para aplicações que utilizam PostgreSQL, há uma proposta de rotina de auditoria de DML em https://wiki.postgresql.org/wiki/Audit_trigger_91plus .

[7] Para aplicações que utilizam Hibernate, é possível utilizar “Envers” https://docs.jboss.org/envers/docs/ ou outro *event listener*.

- Em sistemas web desenvolvidos em Java, um Filtro[8] pode interceptar as requisições feitas à aplicação.

Mínimo

- Deve-se definir no documento de especificação de requisitos do sistema quais informações deverão ser registradas e o local de armazenamento dos dados da auditoria.

Padrão

- Deve-se definir no documento de especificação de requisitos do sistema quais são as políticas de retenção (tempo mínimo de armazenamento dos dados de auditoria) e de revisão dos logs --- *ie*, procedimentos para revisar os logs, analisando se não há indícios de operações indevidas no sistema.

[8] Mais informações sobre filtros em http://www.oracle.com/technetwork/java/filters-137243.html .

PREVENÇÃO, REAÇÃO E MITIGAÇÃO DE FALHAS DE SEGURANÇA

Esta seção apresenta diretrizes para a realização de procedimentos que garantam uma reação adequada à ocorrência de falhas de segurança. Detalha-se o emprego de backups, testes e tratamento de ocorrências.

Backups

Diretivas para a elaboração de processos e procedimentos envolvendo a construção, uso e manutenção de backups.

Observação. A adequação às diretrizes de backup depende, muitas vezes, de políticas e atuação da área de infraestrutura, mas são importantes aspectos a serem considerados e monitorados no desenvolvimento de aplicações seguras.

Mínimo

- Deve-se incluir no plano de projeto a especificação da necessidade e a atribuição da responsabilidade de realização de backups do banco de dados e dos códigos-fonte do sistema, bem como as políticas de acesso a este backup.

Padrão

- Deve-se definir um procedimento estruturado para a restauração de backups.

- Deve-se definir e capacitar responsáveis pela recuperação dos backups.

Forte

- Deve-se criar *baselines* das versões do sistema, facilitando a recuperação ágil para uma versão anterior.

- Deve-se realizar simulações de restauração de dados continuamente.

Testes

Diretivas para a elaboração de processos e procedimentos para a elaboração, execução e verificação de procedimentos para testes de segurança. No que diz respeito à segurança, é desejável a preocupação com a definição de testes capazes de encontrar vulnerabilidades no sistema, permitindo, assim, que sejam realizadas as devidas correções para evitar que as vulnerabilidades cheguem ao ambiente de produção.

Mínimo

- Deve-se realizar testes manuais de segurança antes de cada versão do *software* que modifique sua estrutura (telas de login, serviços não autenticados, novos formulários com interação com o usuário, *etc*.).

Padrão

- Deve-se garantir, através de testes automatizados, que os serviços e dados sigilosos estão protegidos e disponíveis apenas para os usuários detentores das informações.

- Deve-se elaborar uma política de testes, automatizados ou não, visando a garantia de não vulnerabilidade aos principais ataques conhecidos em sistemas.

- Deve-se definir cenários de testes voltados à garantia dos requisitos não funcionais do *software*, preferencialmente realizado por uma equipe de testes diferente da equipe de desenvolvimento do *software*, com intuito de se evitar vícios.

- Deve-se definir cenários de testes, principalmente nos aspectos de segurança, para os casos de atualizações na arquitetura do sistema (servidores de aplicação, banco de dados, versões de browser, versões de sistema operacional, *etc*.).

Forte

- Deve-se propor constantes desafios entre as equipes para testar segurança dos sistemas em formato de competição.

Ocorrências

Diretivas para construção de procedimentos visando o saneamento e mitigação da ocorrência de falhas de segurança. Quando da ocorrência de uma falha de segurança, é necessária ação imediata para mitigar tal ocorrência e prevenir novas falhas.

Mínimo

- Deve-se manter procedimento planejado para imediata indisponibilização do sistema e realização de manutenção corretiva.

Padrão

- Deve-se definir uma política de acompanhamento pós-correção de ocorrências de falha de segurança.

Forte

- Deve-se utilizar lições aprendidas nas ocorrências passadas para revisar a política de testes e incrementar segurança dos sistemas.

AMBIENTE DE DESENVOLVIMENTO

Esta seção apresenta diretrizes para a instalação, configuração e gerenciamento de ambientes de desenvolvimento de sistemas.

Acesso ao Código-Fonte

Diretivas para controle de acesso dos desenvolvedores ao código-fonte das aplicações.

Observação. Quanto ao sigilo do código-fonte dos sistemas desenvolvidos, devem ser, por padrão, de livre acesso aos servidores da SETIC. As demais situações deverão ser analisadas, projeto a projeto, pela chefia.

Mínimo

- Deve-se utilizar um sistema de controle de versão com controle de acesso e recuperação em caso de falhas. *Eg*, SVN[9].

Padrão

- Deve-se utilizar um controle de versão distribuído, que mantém um repositório completo em cada máquina de desenvolvimento, *Eg*, Git[10], Mercurial[11].

Separação de Ambientes

Diretivas para a separação de ambientes de desenvolvimento/testes/homologação (DESENV/TEST/HML) do ambiente de produção (PRD).

Observação. As aplicações desenvolvidas devem considerar o uso de repositórios seguros de dados, especialmente na forma de banco de dados com acesso controlado ou arquivos criptografados, quando o uso de banco de dados não for possível/desejável.

Observação. Caso seja necessário o envio de e-mails pelas aplicações desenvolvidas, mas sem a exigência de autenticação, utilizar os servidores disponibilizados pelo Coordenadoria de Infraestrutura Tecnológica, com e-mails criados especialmente para o sistema e/ou função do e-mail.

Mínimo

- Deve-se utilizar bancos de dados distintos para cada ambiente.

- Deve-se utilizar servidores de aplicação/web distintos para cada ambiente.

- Deve-se prover acesso ao ambiente de desenvolvimento/testes/homologação apenas aos integrantes da equipe de desenvolvimento e aos interessados no projeto (*stakeholders*).

---

[9] Fonte: https://subversion.apache.org .
[10] Fonte: https://git-scm.com/ .
[11] Fonte: https://www.mercurial-scm.org/ .

Padrão

- Deve-se prover um instalador expresso para a instalação do ambiente necessário para a execução de uma dada aplicação.

- Deve-se realizar testes periódicos para assegurar a segurança do ambiente de desenvolvimento/testes/homologação.

Forte

- Não se deve fornecer as senhas de acesso ao ambiente de produção aos desenvolvedores.

PARAMETRIZAÇÃO PARA PROTEÇÃO DE DADOS

Esta seção apresenta diretrizes para a configuração de proteção a dados sensíveis. São detalhados parâmetros para criptografia, hash e gerenciamento de senhas.

Criptografia e Hash

Diretrizes para a configuração e utilização de algoritmos de criptografia e hash visando prover confidencialidade a dados.

Observação. Dados sigilosos e sensíveis devem ser criptografados sempre que possível. O método de criptografia empregado deve obedecer às particularidades dos dados e de sua utilização, seguindo os parâmetros aqui listados.

Observação. Deve-se utilizar hashes criptográficos sempre que possível, sobretudo nos seguintes casos: verificação da integridade de dados; armazenamento e verificação de senhas; provimento de identificador "único" para objetos em um sistema e geração de números pseudo-aleatórios.

Mínimo

- Deve-se utilizar um método criptográfico que siga o *princípio de Kerckhoffs*[12]; o método de encriptação e seus parâmetros devem ser públicos e estar documentados, somente a chave criptográfica deve ser mantida em sigilo.

- Não se deve utilizar um cifrador que admita um método conhecido para quebra da chave criptográfica melhor do que a força bruta, baseada em tentativa e erro.

- Não se deve utilizar o modo de cifrador de bloco *electronic codebook* (ECB) ou modos menos seguros.

- Não se deve utilizar um tamanho da chave menor que 128 bits (cifrador simétrico) ou 1024 bits (cifrador assimétrico).

- Não se deve utilizar função de *hash* sem algum tipo de *salt*.

Padrão

- Não se deve utilizar algoritmos considerados obsoletos para criptografia e *hash* criptográfico. Exemplos: MD5, SHA1, DES/3DES, RC2, RC4, MD4.

- Não se deve utilizar um tamanho da chave menor que 192 bits (cifrador simétrico) ou 2048 bits (cifrador assimétrico).

- Não se deve distribuir chaves criptográficas sem a utilização de uma infraestrutura de chave pública e, portanto, sem a utilização de um cifrador *assimétrico*.

[12] Shannon, Claude (4 October 1949). "Communication Theory of Secrecy Systems". Bell System Technical Journal. 28: 662. Retrieved 20 June 2014.

Forte

- Não se deve utilizar um tamanho da chave menor que 256 bits (cifrador simétrico) ou 4096 bits (cifrador assimétrico).

Senhas

Diretrizes para geração, distribuição e uso de senhas em sistemas computacionais. Os fatores examinados para o uso de senhas em *software* desenvolvido de maneira segura são:

- **Geração e parametrização de senhas.** Explana critérios para a escolha de senhas cuja finalidade é dificultar sua quebra por força-bruta ou adivinhação. O principal parâmetro considerado é o comprimento (tamanho) da senha. Inclui procedimentos para testes de força de senhas.
- **Armazenamento e distribuição de senhas.** Explana métodos para armazenamento seguro de senhas geradas tanto no lado validado (usuário, programa cliente, *etc.*) quanto no lado validador (*software*, sistema autenticador, *etc*.). Inclui métodos para a transmissão segura de senhas via rede e parâmetros para os procedimentos de mudança de senhas.
- ***Interface.*** Explana medidas necessárias à *interface* para as tentativas de validação de senhas. Inclui parametrização para determinar a frequência de tentativas permitidas para validação de uma senha e a apresentação da senha parcialmente submetida para o usuário.

Mínimo

- **Tamanho da senha.** Não se deve utilizar senhas com menos de 8 caracteres.
- **Variação de símbolos.** Não se deve utilizar somente um tipo de caractere (letras, dígitos e símbolos).
- **Aleatoriedade.** Não se deve utilizar palavras comumente utilizadas para senhas (ou variantes destas), como, por exemplo,
    - nome do animal de estimação, membro da família ou pessoa significativa;
    - datas de aniversário;
    - nome do feriado favorito;
    - algo relacionado ao time esportivo favorito; e
    - as palavras “senha” e “*password*”.
- **Periodicidade de troca.** Não se deve utilizar periodicidade de troca superior a 1 ano.
- **Armazenamento, usuário.** Não se deve armazenar senhas em claro.
- **Armazenamento, *software*.** Deve-se armazenar ao menos o *hash* criptográfico com *salt*[13].
- **Verificação/validação.**
    - Não se deve usar um canal em claro para a transmissão da senha ou elemento correspondente.
    - Não se deve utilizar método de conferência menos seguro que desafios baseados em *hash* ou o uso de *hashes* armazenados.

[13] Florencio et al., An Administrator's Guide to Internet Password Research.

- **Número de tentativas.** Não se deve permitir uma taxa de tentativas de validação de senha superior a 5 tentativas por minuto.
- **Digitação.** Não se deve mostrar diretamente a senha quando esta necessita ser digitada pelo usuário; deve haver opção de habilitar e desabilitar a visualização da senha digitada até então.

Padrão

- **Tamanho da senha.** Não se deve utilizar senhas com menos de 12 caracteres.
- **Variação de símbolos.** Deve-se utilizar pelo menos letras maiúsculas e minúsculas, junto a ao menos um tipo de caractere (dígito, símbolo).
- **Aleatoriedade.** Não se deve elaborar senhas sem auxílio de *software* gerador de senhas aleatórias, configurado para atender aos parâmetros aqui estabelecidos.
- **Testes.** Não se deve utilizar senha que não tenha sido validada por um *software* testador de força de senhas.
- **Periodicidade de troca.** Não se deve utilizar periodicidade de troca de senhas superior a 6 meses.
- **Mudança e recuperação de senha.** Não se deve permitir que se utilize o mesmo canal de validação da senha. Não se deve enviar a senha antiga para o usuário, em claro ou não, sob nenhuma hipótese.
- **Armazenamento, usuário.** Não se deve armazenar senha que não esteja criptografada seguindo o nível *padrão* de criptografia estabelecido neste documento.
- **Número de tentativas.** Não se deve permitir uma taxa de tentativas de validação de senha superior a 5 tentativas por minuto. A senha deve ser bloqueada em caso de no máximo 5 erros de validação consecutivos e sua reabilitação deve depender de processo específico.

Forte

- **Tamanho da senha.** Não se deve utilizar senhas com menos de 20 caracteres.
- **Variação de símbolos.** Deve-se utilizar uma mistura de letras maiúsculas, letras minúsculas, dígitos e símbolos.
- **Testes.** Não se deve utilizar senha que não tenha sido validada por um *software* testador de força de senhas diferente do *software* gerador de senhas.
- **Armazenamento, usuário.** Não se deve armazenar senha que não esteja criptografada seguindo o nível *forte* de criptografia estabelecido neste documento.
- **Verificação/validação.** Deve-se utilizar um método de prova com conhecimento zero de senha[14].
- **Número de tentativas.** Não se deve permitir uma taxa de tentativas de validação de senha superior a 5 tentativas por minuto. A senha deve ser bloqueada em caso de no máximo 5 erros de validação consecutivos e sua reabilitação deve depender de processo específico. Deve-se exigir prova de origem da requisição (*e.g.*, *captchas* para demonstrar que o usuário é humano, assinatura digital para provar que requisição veio do sistema permitido) após a primeira falha.
-

[14] Blum, Manuel; Feldman, Paul; Micali, Silvio (1988). "Non-Interactive Zero-Knowledge and Its Applications". Proceedings of the twentieth annual ACM symposium on Theory of computing (STOC 1988): 103–112.

CICLO DE VIDA DE *SOFTWARE*

Esta seção apresenta diretrizes para reforço da segurança de *software* nas diferentes fases de seu ciclo de vida; projeto, codificação e manutenção. Traz, ainda, diretrizes para a aplicação com as pessoas envolvidas nestas diferentes fases.

Projeto

Diretrizes para tornar seguras práticas utilizadas durante a etapa de elaboração de projeto de sistemas, inserida dentro da metodologia de desenvolvimento do TRT4.

Observação. As práticas aqui elencadas são consonantes com o que consta na portaria que institui a política de segurança da informação neste Tribunal[15].

- Deve-se empregar modelo de projeto de *software* que contemple
    - etapa de modelagem de ameaças;
    - definição clara dos riscos de segurança; e
    - nível de severidade que o comprometimento de dados sensíveis traria ao sistema e à instituição.

- Não se deve omitir, durante o projeto de desenvolvimento de sistema e sua execução, a definição de responsabilidades pela segurança de dados do sistema e como essa responsabilidade será verificada.

- Deve-se utilizar cronograma de projeto que contemple pontos de verificação de segurança do sistema desenvolvido ao longo de sua construção.

Codificação

Diretrizes para tornar seguras práticas utilizadas durante a etapa de codificação de sistemas.

- Deve-se documentar, inclusive no código da aplicação, as medidas protetivas aplicadas no código-fonte, de modo a indicar precisamente o procedimento utilizado e suas peculiaridades.

Manutenção

Diretrizes para tornar seguras práticas utilizadas durante a etapa de manutenção de sistemas.

- Não se deve habilitar as atualizações automáticas de *software* ou componentes utilizados na construção de um sistema, sob pena de introdução indevida de falhas de segurança.

---

[15] Tribunal Regional do Trabalho da 4a Região (2008). “Portaria No 4.772 de 23 de setembro de 2008”. Anexo 06.

- Não se deve modificar *software* de terceiros, salvo quando *estritamente* necessário; controles de segurança internos podem ser invalidados. A mudança deve ser feita pelo desenvolvedor original do sistema sempre que possível.

Pessoal

Diretrizes para a perpetuação de práticas de desenvolvimento seguro junto às pessoas que operam as diferentes fases do ciclo de vida do *software*.

- Deve-se proporcionar treinamento e capacitação de programadores para aquisição e revisão de princípios de segurança computacional e desenvolvimento de *software* seguro.

REVISÕES

REVISÃO, INTEGRAÇÃO E CONTINUIDADE DO DOCUMENTO

Este guia tem periodicidade de revisão ao menos anual.

Com o objetivo de orientar os futuros tópicos a serem explorados --- *sem limitá-los* --- o documento possui um *roadmap* de questões, modificações, revisões e adições levantadas que podem ser abordados nas próximas revisões. Dada sua natureza, o *roadmap* deve ser mantido sempre atualizado, devendo ser revisado a cada nova versão do documento.

As subseções do *roadmap* descrevem agrupamentos lógicos dos conceitos, mas *não* são fixas; elas devem ser adaptadas ao tópicos que abrigam a cada revisão.

Roadmap

Estrutura do Documento e Classificação

1. Rever categorização de produtos e processos, incluindo cada capítulo em uma das categorias:
    - **Produto**. Requisito de segurança de *software*.
    - **Processo**. Práticas no desenvolvimento.
2. Criar seção para catálogo de tecnologias obsoletas em função de deficiências conhecidas de segurança. *E.g.*, *RichFaces*.
3. Propor classificação dos sistemas de acordo com a sua criticidade; dados armazenados, escopo, exposição, *etc.* Observar que alguns sistemas não aderem a essa classificação, citando exemplos.

Incidentes de Segurança

1. Documentar incidentes mais comuns no CERT, avaliando os níveis de ameaça de maneira uniforme.
2. Adicionar discussão sobre gestão de incidentes:
    - medidas de contenção caso a segurança de um sistema tenha sido comprometida;
    - procedimentos a serem adotados em caso de incidente de segurança.

Armazenamento de Dados

1. Analisar o uso de lista de *one-time pads* predefinidas para sistemas.

Auditoria

1. Discutir proteção de acesso, proteção contra alterações não-autorizadas, verificação de integridade.
    - Na gravação no banco de dados, utilizar usuário restrito, que possua somente permissão de inserção, sem a possibilidade de alterar ou excluir dados.

- Os sistemas deve pode, ser necessário, permitir selecionar diferentes níveis de *logs*.

Autenticidade

1. Discutir controle de autenticidade dos emails utilizados pelos sistemas.

Atualizações

1. Discutir política de atualização de sistemas com falhas críticas de segurança.

Projeto e Desenvolvimento de *Software*

1. Verificar o campo de “riscos não-funcionais” no processo de desenvolvimento de *software* para inserir riscos de segurança como fatores a serem elencados.

Ambiente de Desenvolvimento e Tecnologias

1. Prover orientações gerais para melhorar a segurança dentro do ambiente de desenvolvimento.
2. Discutir diretrizes de segurança para as linguagens de programação utilizadas.
3. Estabelecer critérios para a avaliação da segurança de ambientes de desenvolvimento. *E.g.*,
    - sensibilidade dos dados manipulados;
    - políticas de sigilo interno e externo de artefatos dos sistema;
    - controles de segurança já existentes; e
    - confiabilidade do pessoal e restrições.
4. Discutir grau de acesso ao ambiente de desenvolvimento de pessoal terceirizado associado. *E.g.*,
    - formalizar acesso e medidas protetivas em contrato, incluindo práticas de segurança que devem ser adotadas pelo terceirizado durante o desenvolvimento;
    - fornecer um modelo de ameaça para o desenvolvedor externo;
    - elaborar testes, com respeito à segurança, dos itens entregues; e
    - discutir uso de acordos de garantia.

Controle de Usuários

1. Listar e explanar parâmetros para utilização de usuários de consulta: efetiva utilização, regulação, etc.

Monitoramento de Vulnerabilidades

1. Acompanhamento e atualização periódica de relatórios sobre vulnerabilidades de segurança prementes, como o da OWASP[16].

Senhas

2. Discutir o uso de gerenciadores e armazenadores de senhas. *Eg*, *Passbolt*, KeyPass, *etc.*Fatores considerados:
    - necessidade da ferramenta ser *open source*;
    - possibilidade de instalação de servidores próprios;
    - armazenamento criptografado das senhas; e
    - compartilhamento de senhas entre times.

[16] https://www.owasp.org

Alternativamente, discutir construção de uma ferramenta própria que atenda às necessidades internas.

Testes

1. Discutir a ampla utilização de testes de segurança, manuais e automatizados, de diferentes tipos, em ambiente separado de homologação e produção.
2. Discutir testes de aceitação independentes, com uso de *scanners* de vulnerabilidade em testes regulares e proteção dos dados utilizados para testes.
3. Discutir tratamento de casos de “abuso de segurança” em baterias de testes automatizados ou manuais.
4. Discutir a prevenção de ataques de *phishing*.
5. Discutir a prevenção de ataques utilizando páginas falsas.
6. Discutir a prevenção de ataques utilizando *cookies* de terceiros.
7. Detalhar principais ataques em sistemas.

GLOSSÁRIO

ATAQUES E DEFESAS

***SQL Injection*.** É uma forma de ataque em sistemas, realizado via interface, no qual o usuário informa trechos de SQL em campos de texto (ou até mesmo em telas de login ou de pesquisa), alterando a consulta prevista pelo desenvolvedor, sendo que o atacante poderá receber privilégios especiais ou poderá manipular indevidamente o banco de dados[17].

CRIPTOGRAFIA

Encriptação. É o processo de converter texto em claro em texto cifrado.

Decriptação. É o processo de converter texto cifrado em seu texto em claro original.

Cifrador. É um par de algoritmos que realizam a encriptação e a decriptação.

Chave. É uma sequência de bits utilizada como parâmetro secreto no cifrador, necessária para realizar encriptação e/ou decriptação. A única maneira de descobrir uma chave deve ser por *força-bruta*; tentar todas as alternativas no espaço de chaves possíveis. O algoritmo utilizado pelo cifrador deve garantir que chaves longas implicam em um tempo impraticável para descobrir a chave por tentativa-e-erro.

Cifrador Simétrico. É um cifrador que usa a mesma chave para encriptação e decriptação. Mais rápidos, em geral. Exemplos: *Advanced Encryptation Standard* (AES)[18] e *Data Encryptation Standard* (DES)[19].

Cifrador Assimétrico. É um cifrador que usa chaves diferentes, uma *pública*, uma *privada*, para encriptação e decriptação. Mais lentos, em geral, mas com usos para assinatura e verificação de autenticidade. Exemplos: Rivest-Shamir-Adleman (RSA)[20] e *Elliptic Curve Cryptography* (ECC)[21].

---

[17] Mais informações sobre *SQL Injection* podem ser encontrados em [https://www.owasp.org/index.php/SQL_Injection]

[18] Joan Daemen, Steve Borg e Vincent Rijmen, "The Design of Rijndael: AES - The Advanced Encryption Standard." Springer-Verlag, 2002. ISBN 3-540-42580-2.

[19] National Bureau of Standards, Data Encryption Standard, FIPS-Pub.46. National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.

[20] Rivest, R.; Shamir, A.; Adleman, L. (February 1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Communications of the ACM. 21 (2): 120–126. doi:10.1145/359340.359342.

[21] Koblitz, N. (1987). "Elliptic curve cryptosystems". Mathematics of Computation. 48 (177): 203–209. JSTOR 2007884. doi:10.2307/2007884

***Hash* Criptográfico.** É uma função matemática que mapeia uma entrada de tamanho arbitrário, em *bits*, para um saída de tamanho fixo e que é utilizada para criptografia. A função também é de “mão única”, no sentido de que é impossível inverter-la. A função deve ser determinística, de rápida computação e de alta entropia.

***Salted Hash*.** Fragmento adicionado ao conteúdo original do hash para que a saída mude mesmo que o conteúdo original seja o mesmo.

Cifrador de bloco. Cifrador que que opera sobre blocos de bits de tamanho fixo com uma transformação invariável que é especificada por uma chave simétrica.

AMBIENTE DE DESENVOLVIMENTO

***Open Relay*.** Os servidores de correio eletrônico são classificados como Open Relay quando ele processa um e-mail onde o remetente e o destinatário não são usuários do servidor em questão.

COMUNICAÇÃO SEGURA

WS-ReliableMessaging. Protocolo para entrega segura de mensagens SOAP.

SOAP. Protocolo para troca de mensagens em formato XML.

REST. Protocolo para comunicação entre sistemas utilizando os métodos do protocolo HTTP.