Guia de Requisitos Mínimos de Privacidade e Segurança da Informação para Aplicações Web

PROGRAMA DE PRIVACIDADE E SEGURANÇA DA INFORMAÇÃO (PPSI)

Versão 2.0

Brasília, abril de 2022

GUIA REQUISITOS MÍNIMOS DE PRIVACIDADE E SEGURANÇA DA INFORMAÇÃO PARA APLICAÇÕES WEB

MINISTÉRIO DA GESTÃO E DA INOVAÇÃO EM SERVIÇOS PÚBLICOS

Esther Dweck

Ministra

SECRETARIA DE GOVERNO DIGITAL

Rogério Souza Mascarenhas

Secretário de Governo Digital

DIRETORIA DE PRIVACIDADE E SEGURANÇA DA INFORMAÇÃO

Leonardo Rodrigo Ferreira

Diretor de Privacidade e Segurança da Informação

COORDENAÇÃO-GERAL DE PROTEÇÃO DE DADOS

Loriza Andrade Vaz de Melo

Coordenadora-Geral de Proteção de Dados

Equipe Técnica de Elaboração

Fábio Hitsuki Nitto

Luiz Henrique do Espírito Santo Andrade

Marcus Paulo Barbosa Vasconcelos Tássio

Correia da Silva

Equipe Revisora

Marcelo de Lima

Equipe Técnica de Revisão - Versão 2.0

Adriano de Andrade Moura

Francisco Magno Felix Nobre

Leonard Keyzo Yamaoka Batista

Rogério Vinícius Matos Rocha

Histórico de Versões

Data	Versão	Descrição	Autor
12/04/2021	1.0	Primeira versão do Guia de Segurança em Aplicações Web.	Equipe Técnica de Elaboração
15/04/2023	2.0	Atualização para alinhamento com o Guia do Framework de Privacidade e Segurança da Informação	Equipe Tecnica de Revisão

Sumário

AVISO PRELIMINAR E AGRADECIMENTOS

O presente Guia, especialmente recomendado e dirigido aos órgãos e às entidades da Administração Pública Federal - APF, visa a auxiliar na especificação de Requisitos Mínimos de Privacidade e Segurança da Informação para Web, em atendimento ao previsto no Capítulo VII - DA SEGURANÇA E DAS BOAS PRÁTICAS Lei nº 13.709, de 14 de agosto de 2018 - Lei Geral de Proteção de Dados Pessoais (LGPD), que determina que a Administração Pública, ao prestar diversos serviços que tratam dados pessoais à sociedade, deve adotar medidas de segurança, técnicas e administrativas aptas a proteger os dados pessoais de acessos não autorizados e de situações acidentais ou ilícitas de destruição, perda, alteração, comunicação ou qualquer forma de tratamento inadequado ou ilícito dos dados que estão sob sua custódia, bem como assegurar o mínimo de segurança nas interfaces de programação de aplicação. Adicionalmente, a especificação de Requisitos Mínimos de Privacidade e Segurança da Informação para Web visa a atender, além da LGPD, a outros normativos vigentes sobre o tema de privacidade e segurança da informação.

Este documento é de autoria exclusiva da Secretaria de Governo Digital (SGD) do Ministério da Gestão e da Inovação em Serviços Públicos e tem como referência fundamental o Guia do Framework de Privacidade e Segurança da Informação baseado em diversas publicações e documentos técnicos já existentes que são utilizados amplamente por profissionais da área de privacidade e segurança da informação. Destacam-se as publicações do Center for Internet Security (CIS), da International Organization for Standardization (ISO), do National Institute of Standards and Technology (NIST). Em complemento ao Guia do Framework de Privacidade e Segurança da Informação, este **Guia** foi inspirado em publicações da Open Web Application Security Project (OWASP). Com o objetivo de facilitar a difusão de conhecimentos sobre privacidade e segurança da informação, tais referências, quando escritas em línguas estrangeiras, foram traduzidas para o português pela equipe técnica da Diretoria de Privacidade e Segurança da Informação da Secretaria de Governo Digital.

Nesse cenário, a Secretaria de Governo Digital enfatiza que:

a) não representa, tampouco se manifesta em nome do CIS, da ISO do NIST e da OWASP e vice-versa;

b) não se manifesta em nome de autoridades de privacidade e segurança da

informação;

c) não é coautora das publicações internacionais abordadas;

d) não assume nenhuma responsabilidade administrativa, técnica ou jurídica por usos ou interpretações inadequadas, fragmentados ou parciais do presente guia; e

e) caso o leitor deseje se certificar de que atende integralmente os requisitos das publicações das instituições mencionadas, deverá consultar diretamente as fontes oficiais de informação ofertadas por elas, que foram listadas na seção "Referências Bibliográficas" **deste documento**.

Finalmente, um agradecimento especial deve ser registrado ao CIS, à ISO, ao NIST, ao OWASP e aos profissionais de privacidade e segurança da informação consultados, por suas valiosas contribuições para a comunidade e para elaboração **deste documento**.

Este Guia será atualizado frequentemente, de acordo com as novas diretrizes determinadas pelas autoridades em privacidade e segurança da informação ou segundo eventuais alterações que ocorram nos normativos vigentes relacionados a privacidade e segurança da informação e outras referências utilizadas **neste documento**.

INTRODUÇÃO

Este Guia tem por finalidade apresentar orientações com o intuito de auxiliar os órgãos e entidades da Administração Pública Federal, direta, autárquica e fundacional a especificar os Requisitos Minimos de Privacidade e Segurança da Informação no âmbito institucional.

O Controle 16 (p. 55) e 22 (p. 63) do Guia do Framework de Privacidade e Segurança da Informação, estabelece que:

Controle 16: Segurança de Aplicações *- Gerenciar o ciclo de vida de segurança de todos os softwares desenvolvidos e adquiridos internamente, a fim de prevenir, detectar e corrigir falhas de segurança.*

Controle 22: Políticas, Processos e Procedimentos *- Definir, desenvolver, divulgar, implementar e atualizar políticas, processos e procedimentos operacionais, internos e externos que regem as ações relativas à proteção de dados pessoais e privacidade, e controles para programas, sistemas de informação ou tecnologias que envolvam o tratamento de dados pessoais.*

O presente Guia serve como um modelo prático a ser utilizado para auxiliar na adoção de medidas dos Controles 16 e 22 do Guia do Framework de Privacidade e Segurança da Informação[1] v1 e respectivas evoluções desta versão (1.1, 1.2 etc.) elaborado e publicado pela SGD. As medidas do Controles 16 e 22 que estão contempladas por este Guia são: 16.1, 16.2, 16.3, 16.4, 16.5, 16.6, 16.7, 16.8, 16.9, 16.10, 16.11, 16.12, 16.13, 16.14 e 22.7,

Recomenda-se que a instituição observe o previsto na Norma Complementar nº 16 DSIC/GSIPR, de 21/11/12, que trata de diretrizes para o Desenvolvimento e Obtenção de Software Seguro nos Órgãos e Entidades da Administração.

A abordagem oferece uma visão geral dos principais requisitos e orientações relacionadas ao desenvolvimento seguro, e estas proposições não devem ser vistas como uma referência completa e suficiente para o tema, tampouco substituir a necessidade de a instituição compreender sua própria postura de risco institucional.

[1] < https://www.gov.br/governodigital/pt-br/seguranca-e-protecao-de-dados/ppsi/guia_framework_psi.pdf >. Acesso: 10 abr. 2023

1 Diretrizes gerais

As diretrizes gerais, para atingir as melhores práticas no desenvolvimento seguro, são embasadas no Guia do Framework de Privacidade e Segurança da Informação e consistem nas medidas contidas no controle 16, que auxiliam a detecção, a resposta e a mitigação de danos provocados por esses ataques, e na medida 22.7 do controle 22, que rege procedimentos operacionais relativos à proteção de dados pessoais e privacidade durante o desenvolvimento de software. . Tais medidas estão expostas abaixo:

1. Estabelecer e manter um processo de desenvolvimento de aplicações seguro. Este processo deve tratar de itens como padrões de design seguro de aplicações (Security by Design), práticas de codificação seguras, treinamentos para desenvolvedores, gestão de vulnerabilidades, segurança de código de terceiros e procedimentos de teste de segurança de aplicação. Deve ser realizada uma revisão e/ou alteração deste processo periodicamente, em casos específicos ou quando ocorrerem mudanças na organização que venham impactá-la de forma significativa.
2. Estabelecer e manter um processo de aceitação e tratamento de informações sobre vulnerabilidades de softwares, incluindo mecanismos para que entidades externas contatem o grupo de segurança da instituição. É importante que o processo inclua itens como: Política de tratamento de vulnerabilidades identificadas e relatadas, equipe ou profissional responsável por analisar os relatórios de vulnerabilidade e um processo de entrada, atribuição, correção e testes de correção. Como parte deste processo, é importante rastrear as vulnerabilidades, classificar a gravidade e atribuir métricas capazes de medir o tempo de identificação, análise e correção das vulnerabilidades. Deve ser realizada uma revisão e/ou alteração deste processo periodicamente, em casos específicos ou quando ocorrerem mudanças na organização que venham impactá-la de forma significativa.
3. Executar a análise de causa raiz em vulnerabilidades de segurança. A análise da causa raiz é a tarefa capaz de avaliar os problemas subjacentes que criam vulnerabilidades no código da aplicação e permite que as equipes de desenvolvimento vão além de apenas corrigir vulnerabilidades individuais

conforme elas surgem.

4. Estabelecer e gerenciar um inventário atualizado de componentes de terceiros usados no desenvolvimento, geralmente chamados de “lista de materiais”, bem como componentes programados para uso futuro. Este inventário deve incluir quaisquer riscos que cada componente de terceiros possa representar a organização. Deve ser realizada uma revisão e/ou alteração deste inventário periodicamente, com o objetivo de identificar quaisquer mudanças ou atualização nesses componentes e validar a compatibilidade do mesmo.
5. Utilizar apenas componentes de terceiros atualizados e confiáveis. Quando possível, escolher bibliotecas e estruturas pré-estabelecidas e comprovadas que forneçam a segurança adequada. É importante adquirir tais componentes de fornecedores e fontes confiáveis ou realizar a avaliação de vulnerabilidades do software antes de usar/adquirir.
6. Estabelecer e manter um processo para a classificação de gravidade de vulnerabilidades capaz de facilitar a priorização na medida em que as vulnerabilidades descobertas são corrigidas. Esse processo deve incluir a definição de um nível mínimo de aceitabilidade de segurança para a liberação de código ou aplicações. A classificação de gravidade deve trazer uma forma sistemática de triagem de vulnerabilidades que venha a melhorar a gestão de riscos e ajuda a garantir que os bugs mais graves sejam priorizados. Revise o processo o e a classificação de vulnerabilidade periodicamente.
7. Utilizar modelos de configuração de segurança padrão (Segurança by Default) recomendados pela equipe de segurança em componentes de infraestrutura de aplicações. Isso inclui servidores subjacentes, bancos de dados e servidores web e se aplica a contêineres de nuvem, componentes de Platform as a Service (PaaS) e componentes de Security as a Service (SaaS). Não permita que o software desenvolvido internamente enfraqueça as configurações de segurança da organização.
8. Manter ambientes separados para sistemas de produção e não produção.
9. Garantir que todos os responsáveis pelo desenvolvimento de software recebam treinamento para escrever código seguro para seu ambiente de desenvolvimento

e responsabilidades específicas. O treinamento deve incluir princípios gerais de segurança e práticas padrão de segurança para aplicações. O treinamento deve ser realizado periodicamente, é interessante estabelecer uma cultura de segurança entre os desenvolvedores.

10. Aplicar princípios de design seguro em arquiteturas de aplicações. Os princípios de design seguro incluem o conceito de privilégio mínimo e aplicação de mediação para validar cada operação que o usuário faz, promovendo o conceito de "nunca confiar nas entradas do usuário". Os exemplos incluem garantir que a verificação explícita de erros seja realizada e documentada para todas as entradas, incluindo tamanho, tipo de dados e intervalos ou formatos aceitáveis. O design seguro também significa minimizar a superfície de ataque da infraestrutura da aplicação, como desligar portas e serviços desprotegidos, remover programas e arquivos desnecessários e renomear ou remover contas padrão.
11. Aproveitar módulos ou serviços controlados para componentes de segurança da aplicação, como gestão de identidade, criptografia e auditoria de logs. O uso de recursos para plataforma em funções críticas de segurança deve reduzir a carga de trabalho dos desenvolvedores e minimizará a probabilidade de erros de design ou implementação. Os sistemas operacionais modernos fornecem mecanismos eficazes para identificação, autenticação e autorização e disponibilizam esses mecanismos para as aplicações. Use apenas algoritmos de criptografia padronizados, atualmente aceitos e amplamente revisados. Os sistemas operacionais também fornecem mecanismos para criar e manter logs de auditoria seguros.
12. Utilizar ferramentas de análise estáticas e dinâmicas dentro do ciclo de vida da aplicação para verificar se as práticas de codificação seguras estão sendo utilizadas na organização.
13. Realizar testes de invasão em aplicações. Para aplicações críticas, o teste de invasão autenticado é mais adequado para localizar vulnerabilidades de codificação e de negócios do que a varredura de código e o teste de segurança automatizado. O teste de invasão depende da habilidade do testador para manipular manualmente uma aplicação como um usuário autenticado e não

autenticado.

14. Realizar a modelagem de ameaças. A modelagem de ameaças é o processo de identificar e abordar as falhas de design de segurança da aplicação em um desenho, antes que o código seja criado. É conduzido por profissionais especialmente treinados que avaliam o design da aplicação e medem os riscos de segurança para cada ponto de entrada e nível de acesso. O objetivo é mapear a aplicação, a arquitetura e a infraestrutura de uma forma estruturada para entender todos os pontos fracos.
15. Estabelecer procedimento ou metodologia para assegurar que os princípios da LGPD estão sendo respeitados desde a fase de concepção do produto ou do serviço até a sua execução (Privacy by Design).

Reforça-se que as diretrizes têm como objetivo auxiliar a instituição a amadurecer as linhas de defesa de forma gradativa e não impedem que medidas de perfil mais avançado sejam adotadas pela a instituição.

2 Requisitos gerais

Este capítulo trata de requisitos gerais que compreendem a adoção de um processo de gerenciamento de ambiente e a adoção de mecanismos de proteção do perímetro da aplicação.

Os requisitos gerais visam reduzir a exposição dos ativos a vetores de ataque, e compreendem ações como, por exemplo: *privacy by design*, aplicação de *patches* de segurança e atualização de softwares que sustentam a aplicação; eliminação de protocolos, serviços ou portas nos servidores da aplicação que não sejam essenciais para seu funcionamento; limitação da exibição de informações sobre os ativos que sustentam a aplicação; e mitigação e bloqueio de requisições suspeitas direcionadas à aplicação de forma a impossibilitar a execução de ataques.

2.1 Privacidade

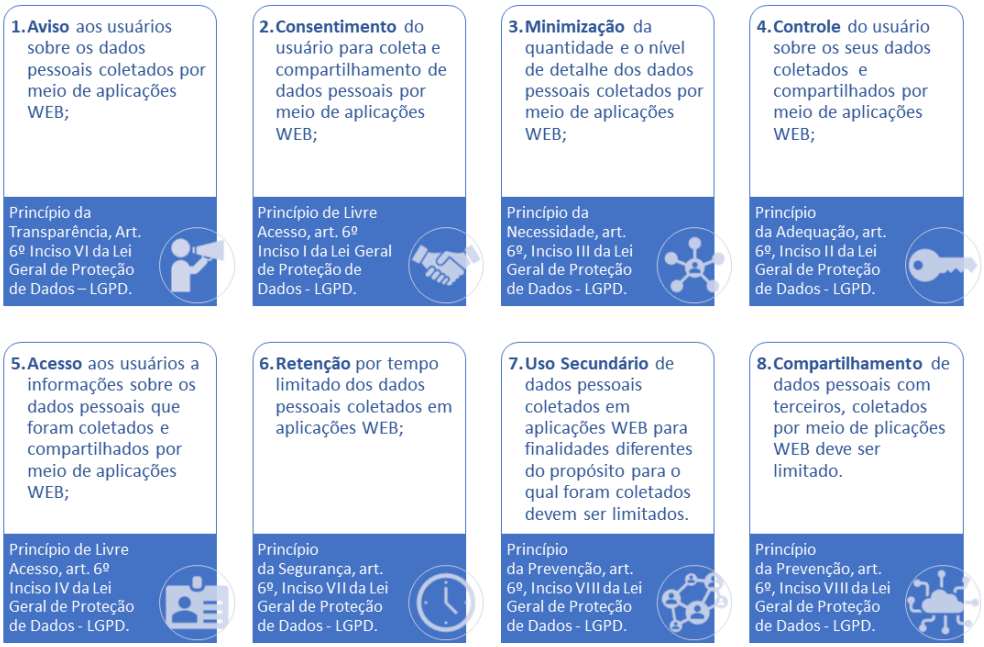
Esta seção descreve boas práticas de privacidade para aplicações web, incluindo aquelas que podem usar APIs (*Application Programming Interfaces*)[2].

2.1.1 Privacy by Design

A ideia de *Privacy by Design* é incorporar medidas protetivas de privacidade e dados pessoais, em todas as fases dos projetos em desenvolvimento. A princípio, não seria permitido desenvolver nenhum projeto, produto ou serviço sem que os princípios de proteção da privacidade estejam no centro desse desenvolvimento. Em outras palavras, o produto ou serviço deve ser lançado e recebido pelo usuário com todas as salvaguardas que foram concebidas durante o seu desenvolvimento. Logo, todas as medidas para proteger a privacidade, idealizadas desde o início do desenvolvimento do projeto, devem atender esse princípio.'

O *Privacy by Design* pode ser mais bem compreendido por intermédio da análise dos princípios basilares de privacidade, segundo a Nota do Grupo de Trabalho W3C[3]. Tais princípios, que são relevantes para as aplicações Web efetuam uma correlação com os princípios da LGPD:

[2] Device API Privacy Requirements - <http://www.w3.org/TR/2010/NOTE-dap-privacy-reqs-20100629/>
[3] Web Application Privacy Best Practices (w3.org) - <https://www.w3.org/TR/2012/NOTE-app-privacy-bp-20120703/>



2.2 Segurança

Esta seção descreve os requisitos gerais de segurança para o desenvolvimento e manutenção de aplicações Web.

2.2.1 Gerenciamento de ambiente

O trabalho da organização na segurança das aplicações não termina quando a aplicação se torna operacional. Novos recursos de segurança e *patches* são lançados regularmente para as várias tecnologias que sustentam a aplicação, até que se tornem obsoletas ou não tenham mais suporte.

Patches de segurança e novas versões são lançadas frequentemente para corrigir as vulnerabilidades que são descobertas ao longo dos ciclos de vida das tecnologias. Por isso, é essencial monitorar as vulnerabilidades das tecnologias utilizadas pela aplicação, aplicar *patches* ordenados e oportunos em todos os sistemas afetados, bem como utilizar versões seguras.

O gerenciamento de ambiente se concentra em manter o ambiente das aplicações seguro pela aplicação de *patches* e da atualização dos softwares que sustentam a aplicação, e por meio do uso de técnicas, configurações, ferramentas e melhores práticas para eliminar potenciais vetores de ataque *(hardening)*[4].

4 https://owaspsamm.org/model/operations/environment-management/

Para que a aplicação de *patches* e atualização dos softwares sejam realizadas de forma eficiente, a organização deve adotar um processo de gerenciamento para identificar os *patches* e atualizações necessárias, planejar essas atividades, documentá-las, executá-las e, por fim, mensurar todo o processo.

Além disso, por meio do *hardening*, busca-se blindar os componentes que sustentam a aplicação, mitigar e corrigir vulnerabilidades, e reduzir a superfície de ataque dos componentes da aplicação.

2.2.2 Proteção do perímetro da aplicação

O perímetro da aplicação consiste na fronteira entre a rede interna da instituição e a Internet – ou outra rede externa.

O perímetro da aplicação inclui: roteadores de borda, Firewalls, sistemas de prevenção e detecção de intrusões (IPS e IDS), e redes de perímetro, como as zonas desmilitarizadas (ZDM).

A proteção do perímetro da aplicação consiste na aplicação de controles e regras para o tráfego que flui entre a fronteira da rede interna e externa. Esses controles de segurança visam bloquear o tráfego de rede suspeito ou malicioso, limitar o acesso às aplicações somente a endereços confiáveis e necessários, registrar informações sobre os pacotes de rede que atravessam a fronteira, de modo a identificar possíveis incidentes de segurança, impedindo-os ou reportando-os ao responsável pela segurança da aplicação. Redes de perímetro também podem ser utilizadas para separar a aplicação de outros serviços, e evitar que um potencial dano à eles possam comprometer a aplicação.

3 Requisitos específicos

Este capítulo traz as melhores práticas de privacidade para aplicações Web, retiradas da Nota "*Web Application Privacy Best Practices*" do Grupo de Trabalho W3C[5], e 14 categorias de requisitos específicos de segurança em aplicações Web extraídos do guia de "Melhores Práticas de Codificação Segura OWASP"[6], tais categorias devem ser analisadas ao longo do ciclo de vida da aplicação. É importante salientar que a listagem abaixo não representa um conjunto de requisitos exaustivos:

3.1 Privacidade

Esta seção descreve as boas práticas (requisitos específicos) de privacidade para o desenvolvimento e manutenção de aplicações Web.

3.1.1 Privacy by Design

A metodologia *Privacy By Design* é uma grande aliada no auxílio à adequação à LGPD, em especial ao artigo 46, que prevê:

> *Art. 46. Os agentes de tratamento devem adotar medidas de segurança, técnicas e administrativas aptas a proteger os dados pessoais de acessos não autorizados e de situações acidentais ou ilícitas de destruição, perda, alteração, comunicação ou qualquer forma de tratamento inadequado ou ilícito.*
>
> *(...)*
>
> *§ 2º As medidas de que trata o caput deste artigo deverão ser observadas desde a fase de concepção do produto ou do serviço até a sua execução.*

Os princípios de *Privacy by Design* devem ser refletidos no design e no desenvolvimento de aplicações Web. Quando adotados, seu objetivo é justamente garantir a proteção dos dados desde a concepção do projeto.

MELHOR PRÁTICA 1: Seguir os princípios de "*Privacy by Design*".

Considere proativamente a privacidade, torne a preservação da privacidade o padrão,

[5] Web Application Privacy Best Practices - <https://www.w3.org/TR/2012/NOTE-app-privacy-bp-20120703/#bib-MWABP>
[6] OWASP Secure Coding Practices-Quick Reference Guide - https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/

incluindo a privacidade em um design centrado ao usuário e transparente, sem fazer compensações contra a privacidade de outros recursos, pois a privacidade é possível junto com outras funcionalidades. São princípios dessa metodologia:

3.1.1.1 Proativo, não reativo; preventivo, não corretivo

A abordagem desse princípio é caracterizada por medidas proativas ao invés de reativas, ela antecipa e evita eventos invasivos de privacidade antes que eles aconteçam. Não espera por riscos de privacidade se concretizem, nem oferece soluções para resolver violações de privacidade uma vez que elas ocorreram. Visa, sim, a impedir que eles ocorram. Em resumo, o *Privacy by Design* vem antes do fato, não depois.

3.1.1.2 Privacidade como configuração padrão

Conhecido também por *Privacy by Default, este* princípio tem como caraterística fundamental oferecer o grau máximo de privacidade, garantindo que os dados pessoais sejam protegidos automaticamente em qualquer sistema. Mesmo que um usuário não faça configuração alguma, sua privacidade ainda assim permanecerá preservada. Em resumo, a privacidade é preservada por padrão no sistema.

3.1.1.3 Privacidade incorporada ao *design*

Este princípio prevê que a privacidade seja incorporada ao *design* e a arquitetura dos sistemas de TI e das práticas de negócios. Isto significa que a privacidade não deve ser acoplada como um complemento após o acontecimento de um incidente. Como resultado, a privacidade se torna um componente essencial no núcleo da funcionalidade que está sendo entregue.

3.1.1.4 Funcionalidade total - Soma positiva, não soma zero

Também conhecido como "ganha-ganha", este princípio é essencialmente sobre como evitar *trade-offs* (escolher uma opção em detrimento de outra): acreditar que, em qualquer sistema ou serviço, é possível apenas ter ou privacidade ou segurança, e não ambos simultaneamente. Dessa forma, devem-se incorporar todos os objetivos legítimos e, ao mesmo tempo, garantir o cumprimento de suas obrigações.

3.1.1.5 Segurança de ponta a ponta - Proteção total do ciclo de vida

Princípio que busca a segurança de ponta a ponta, protegendo os dados coletados durante todo seu ciclo de vida e abrangendo coleta, uso, acesso, armazenamento e descarte. Este princípio parte do pressuposto de que, sem segurança adequada, não existe privacidade. Nesse caso, é preciso que as organizações que fazem parte da APF assumam de fato a responsabilidade pela segurança dos dados pessoais.

3.1.1.6 Visibilidade e transparência - Mantenha-o aberto

O *Privacy by Design* visa garantir a todas as partes interessadas que, seja qual for a prática de negócios ou a tecnologia envolvida, a operação está de acordo com as promessas e os objetivos declarados, sujeitos à verificação independente. Seus componentes, peças e operações permanecem visíveis e transparentes, tanto para usuários quanto para fornecedores. Recorde-se de confiar, mas verificar.

3.1.1.7 Respeito pela privacidade do usuário - Mantenha-o centrado no usuário

Princípio que tem como objetivo manter os interesses do indivíduo em primeiro lugar, com enfoque no respeito à sua privacidade. Devem-se oferecer medidas com configurações fortes de privacidade, informações claras e opções amigáveis.

3.1.2 Design Centrado no Usuário

A privacidade deve ser centrada no usuário, dando-lhe compreensão e controle sobre o uso de seus dados pessoais.

MELHOR PRÁTICA 2: Permitir que o usuário tome decisões informadas sobre o compartilhamento de suas informações pessoais com um serviço.

O usuário final deve ter informações suficientes sobre um serviço e como ele usará suas

informações pessoais para tomar uma decisão informada sobre compartilhar informações com esse serviço. Isso deve incluir a compreensão dos dados a serem compartilhados, clareza sobre por quanto tempo os dados serão mantidos e informações com quem serão compartilhados (e para que propósito).

MELHOR PRÁTICA 3: Permitir que o usuário tome decisões no momento apropriado com as informações contextuais corretas.

O usuário tem a oportunidade de decidir se deseja compartilhar informações (e o que compartilhar), quando for necessário. Isso é importante porque a decisão pode depender do contexto, inclusive no que concerne aos detalhes do que o usuário está tentando realizar, aos detalhes dessa tarefa e às diferenças em como o serviço irá operar, usar e compartilhar dados.

A aplicação Web deve certificar-se de que o consentimento é um "consentimento informado" e fornecer o aviso de privacidade necessário e as demais informações quando o consentimento do usuário for solicitado, por meio de ação ou outros meios.

MELHOR PRÁTICA 4: Ao aprender as decisões de privacidade do usuário e fornecer padrões, deve-se permitir que o usuário visualize e altere facilmente suas decisões anteriores.

Um serviço pode aprender e lembrar informações pessoais do usuário para melhorar a experiência. Exemplo: lembrar as informações de pagamento. O serviço deixa claro para o usuário quais informações são retidas e como são usadas. Deve-se dar ao usuário a oportunidade de corrigir ou remover as informações.

MELHOR PRÁTICA 5: Concentrar-se na usabilidade e evitar solicitações desnecessárias.

O foco na usabilidade deve melhorar um serviço, bem como tornar mais fácil para o usuário entender e controlar o uso de suas informações pessoais. Minimize o uso de diálogos modais (*pop-up*, por exemplo), pois eles prejudicam a experiência do usuário e muitos usuários não entenderão como responder aos prompts, em vez de fazer uma escolha que lhes permita continuar seu trabalho.

MELHOR PRÁTICA 6: O consentimento ativo deve ser dado livremente, para dados específicos, e ser informado.

O consentimento ativo é o momento em que a ação do usuário é realizada para também

dar permissão, evitando a necessidade de diálogos de consentimento. Esse consentimento ativo deve ser dado livremente, para dados específicos, e ser informado. Assim, o usuário deve ser capaz de cancelar a operação, saber quais dados são compartilhados e ter informações adequadas no momento da ação quanto ao uso pretendido dos dados (consentimento). A aplicação Web deve fornecer ao usuário informações sobre o uso pretendido em conjunto com o uso da API do dispositivo.

Exemplo: consentimento ativo inclui a seleção de campos de contato para compartilhar, a escolha de criar uma imagem clicando no obturador da câmera, e assim por diante. O consentimento ativo pode melhorar a usabilidade e ser menos prejudicial que os diálogos de consentimento. Pode igualmente atender aos requisitos de privacidade se os critérios apropriados forem atendidos.

MELHOR PRÁTICA 7: Ser claro e transparente para os usuários em relação a possíveis preocupações com a privacidade.

O usuário final deve entender se as informações estão sendo usadas pelo próprio serviço ou compartilhadas com terceiros, especialmente quando os serviços de terceiros estão envolvidos em um "*mashup*" (uso de conteúdo de mais de uma fonte).

MELHOR PRÁTICA 8: Ser claro se as informações são necessárias apenas uma vez ou por um período, e por quanto tempo.

O usuário final deve entender se as informações coletadas são para um único uso ou serão retidas e terão um impacto ao longo do tempo.

3.1.3 Minimize a Coleta e Transmissão de Dados Pessoais

Revise os dados e como eles são estruturados e usados, minimizando a quantidade e os detalhes de dados necessários para fornecer um serviço.

MELHOR PRÁTICA 9: Solicitar uma quantidade mínima de dados com detalhes mínimos necessários para fornecer um serviço.

Exemplo: uma entrada do catálogo de endereços não é o nível natural de granularidade, pois o usuário pode desejar compartilhar vários campos individuais independentemente. Portanto, o nível natural de granularidade dele seriam os campos necessários a serem fornecidos

em resposta a uma solicitação de entrada do referido catálogo.

MELHOR PRÁTICA 10: Reter a quantidade mínima de dados com mínimos detalhes pelo tempo mínimo necessário. Considere possíveis usos indevidos de dados retidos e possíveis contramedidas.

Exemplo: reter informações de pagamento do usuário acarreta o risco de que essas informações sejam roubadas e utilizadas indevidamente. Talvez não precise ser retido, mas se o for (com permissão do usuário), talvez deva ser criptografado.

3.1.4 Manter a Confidencialidade dos Dados Pessoais

Manter a confidencialidade dos dados do usuário armazenados.

MELHOR PRÁTICA 11: Manter a confidencialidade dos dados do usuário na transmissão, por exemplo, usando HTTPS para transporte seguro ao invés de HTTP.

O uso de HTTPS pode fornecer confidencialidade de dados pessoais no transporte quando um conjunto de cifras apropriado é necessário. Isso deve ser feito para informações pessoais sensíveis, a menos que a confidencialidade possa ser garantida por outros meios.

MELHOR PRÁTICA 12: Manter a confidencialidade dos dados do usuário armazenados.

A confidencialidade das informações pessoais deve ser mantida durante o armazenamento, para evitar perdas inadvertidas ou maliciosas.

Exemplo: invasão de um servidor, roubo de backups ou outras ameaças.

3.1.5 Controle e Registro de Acesso

Controle e registre o acesso aos dados.

MELHOR PRÁTICA 13: Controle e registre o acesso aos dados.

Restringir o acesso às informações por meio da implementação de controles de acesso e respectivos registros (logs).

3.2 Segurança

Os requisitos específicos de segurança a seguir, foram embasados no Guia de Referência Rápida das Melhores Práticas de Codificação Segura OWASP publicado em dezembro de 2022.

3.2.1 Validação dos dados de entrada

As vulnerabilidades baseadas nos dados de entrada podem surgir em qualquer funcionalidade de uma aplicação, e podem estar presentes em praticamente toda tecnologia de uso comum existente.

Uma grande variedade de ataques a aplicações Web se origina ou envolve o envio de dados de entrada inesperados ou especialmente construídos para causar comportamentos inesperados na aplicação. Sendo assim, toda entrada de dados em um sistema deve ser considerada, a princípio, não confiável.

Alguns mecanismos de defesa podem ser utilizados, conforme detalhamento disposto na tabela abaixo:

ID	Detalhamento do Controle de Segurança Crítico
1.1	Efetuar toda a validação dos dados em um sistema confiável – por exemplo, centralizar todo o processo no servidor.
1.2	Identificar todas as fontes de dados e classificá-las como sendo confiáveis ou não. Em seguida, validar os dados provenientes de fontes nas quais não se possa confiar (ex: basede dados, *stream* de arquivos etc.).
1.3	A rotina de validação de dados de entrada deve ser centralizada na aplicação.
1.4	Especificar conjunto de caracteres apropriado, como UTF-8, para todas as fontes de entrada de dados.
1.5	Codificar os dados para um conjunto de caracteres comuns antes da validação (*Canonicalize*).
1.6	Quando há falha de validação, a aplicação deve rejeitar os dados fornecidos.
1.7	Determinar se o sistema suporta conjuntos de caracteres estendidos UTF-8 e, em caso

	afirmativo, validar após efetuar a descodificação UTF-8.
1.8	Validar todos os dados provenientes dos clientes antes do processamento, incluindo todos os parâmetros, campos de formulário, conteúdo das URLs e cabeçalhos HTTP, como, por exemplo, os nomes e os valores dos *Cookies*. Certificar-se, também, de incluirmecanismos automáticos de *postback* nos blocos de código *JavaScript*, *Flash* ou qualquer outro código embutido.
1.9	Verificar se os valores de cabeçalho, tanto das requisições, como das respostas, contêm apenas caracteres ASCII.
1.10	Validar dados provenientes de redirecionamentos. Os atacantes podem incluir conteúdomalicioso diretamente para o alvo do mecanismo de redirecionamento, podendo assimcontornar a lógica da aplicação e qualquer validação executada antes do redirecionamento.
1.11	Validar tipos de dados esperados.
1.12	Validar intervalo de dados.
1.13	Validar o tamanho dos dados.
1.14	Validar, sempre que possível, todos os dados de entrada através de um método baseado em “listas de permissões" (*whitelists*) que utilizem uma lista de caracteres ou expressõesregulares para definirem os caracteres permitidos.
1.15	Se qualquer caractere potencialmente perigoso precisa ser permitido na entrada de dados da aplicação, certificar-se de que foram implementados controles adicionais comocodificação dos dados de saída, *APIs* específicas que fornecem tarefas seguras e trilhas de auditoria no uso dos dados pela aplicação. A seguir, como exemplo de caracteres “potencialmente perigosos”, temos: <, >, ", ', %, (,), &, +, \, \', \".

1.16	Se a rotina de validação padrão não abordar as seguintes entradas, então elas devem serverificadas: a) Verificar bytes nulos (%00) b) Verificar se há caracteres de nova linha (%0d, %0a, \r, \n) c) Verificar se há caracteres “ponto-ponto barra” (../ ou ..\) que alteram caminhos. Nos casos de conjunto de caracteres que usem a extensão UTF-8, o sistema deve utilizar representações alternativas como: %c0%ae%c0%ae/. A canonicalização deve ser utilizada para resolver problemas de codificação dupla (*double encoding*) ou outras formas de ataques por ofuscação.

3.2.2 Codificação de dados de saída

Com a diversidade de arquiteturas modernas de aplicações Web, a realização da codificação de saída é muito importante. Pode ser difícil fornecer validação de entrada robusta em certos cenários. Portanto, o uso de APIs mais seguras com consultas parametrizadas, estruturas de modelagem com escape automático ou codificação de saída cuidadosamente escolhida é fundamental para a segurança da aplicação.

O propósito da codificação de saída é converter a entrada não confiável em uma forma segura, onde a entrada é exibida como dados para o usuário, sem executar uma codificação no navegador. A tabela a seguir detalha uma lista de medidas de codificação de saída.

ID	Detalhamento do Controle de Segurança Crítico
2.1	Efetuar toda a codificação dos dados em um sistema confiável - por exemplo, centralizartodo o processo no servidor.
2.2	Utilizar uma rotina padrão e testada para cada tipo de codificação de saída.
2.3	Realizar a codificação, baseada em contexto, de todos os dados enviados para o clienteque têm origem em um ambiente fora dos limites de confiança da aplicação. Acodificação das entidades HTML é um exemplo, mas nem sempre funciona para todos os casos.

2.4	Codificar todos os caracteres, a menos que sejam conhecidos por serem seguros para ointerpretador de destino.
2.5	Realizar o tratamento (sanitização), baseado em contexto, de todos os dados provenientes de fontes não confiáveis usados para construir consultas SQL, XML e LDAP.
2.6	Tratar todos os dados provenientes de fontes que não sejam confiáveis que geremcomandos para o sistema operacional.

3.2.3 Autenticação e gerenciamento de credenciais

Um requisito necessário em praticamente toda aplicação é o acesso do usuário aos seus dados e às funcionalidades. Esse acesso é gerenciado normalmente por três mecanismos interrelacionados:

- Autenticação
- Gerenciamento de sessões (requisito 4)
- Controle de Acesso (requisito 5)

A autenticação é o processo que busca verificar a identidade digital de uma entidade de um sistema quando tal entidade requisita acesso a esse sistema. O processo é realizado por meio de regras preestabelecidas, geralmente pela comparação das credenciais apresentadas pela entidade com outras já pré-definidas no sistema, reconhecendo como verdadeiras ou legítimas as partes envolvidas em um processo. Vide abaixo uma lista de medidas a serem considerados nessa hipótese:

ID	Detalhamento do Controle de Segurança Crítico
3.1	Requerer autenticação para todas as páginas e recursos, exceto para aqueles que sãointencionalmente públicos.
3.2	Os controles de autenticação devem ser executados em um sistema confiável - porexemplo, centralizar todo o processo no servidor.

3.3	Sempre que possível, estabelecer e utilizar serviços de autenticação padronizados etestados.
3.4	Utilizar uma implementação centralizada para realizar os procedimentos de autenticação, disponibilizando bibliotecas que invoquem os serviços externos de autenticação.
3.5	Separar a lógica de autenticação do recurso que está a ser requisitado eusar redirecionadores dos controladores de autenticação centralizados.
3.6	Quando ocorrerem situações excepcionais nos controles de autenticação, executarprocedimentos em caso de falha com o propósito de manter o sistema seguro.
3.7	Todas as funções administrativas e de gerenciamento de contas devem ser tão segurasquanto o mecanismo de autenticação principal.
3.8	Se a aplicação gerenciar um repositório de credenciais, esta deverá garantir que as senhas são armazenadas na base de dados somente sob a forma de resumo/hash da senha na forma de *one-way salted hashes* e que a tabela/arquivo que armazena as senhas e as próprias chaves são manipuladas apenas pela aplicação. Não utilizar algoritmos de *hash* reconhecidamente inseguros.
3.9	A geração dos resumos (*hash*) das senhas deve ser executada em um sistema confiável - por exemplo, centralizar o controle no servidor.
3.10	Validar os dados de autenticação somente no final de todas as entradas de dados,especialmente para as implementações de autenticação sequencial.
3.11	As mensagens de falha na autenticação não devem indicar qual parte dos dados de autenticação está incorreta. Por exemplo, em vez de exibir mensagens como “Nome de usuário incorreto” ou “Senha incorreta”, utilize apenas mensagens como: “Usuário e/ou senha inválidos”, para ambos os casos de erro. As respostas de erro devem ser idênticas nos dois casos.
3.12	Utilizar autenticação para conexão a sistemas externos que envolvam tráfego de informação sensível ou acesso às funções.

3.13	As credenciais de autenticação para acessar serviços externos à aplicação devem ser cifradas e armazenadas em um local protegido de um sistema confiável - por exemplo,no servidor da aplicação. Obs.: o código-fonte não é considerado um local seguro.
3.14	Utilizar apenas requisições *POST* para transmitir credenciais de autenticação.
3.15	Somente trafegar senhas (não temporárias) através de uma conexão protegida (SSL/TLS) ou no formato de dado cifrado, como no caso de envio de e-mail cifrado. Senhas temporárias enviadas por e-mail podem ser um caso de exceção aceitável.
3.16	Exigir que os requisitos de complexidade de senha estabelecidos pela política ou regulamento sejam cumpridos. As credenciais de autenticação devem resistir a ataquesque, tipicamente, ameaçam o ambiente de produção. Um exemplo pode ser a exigênciado uso simultâneo de caracteres alfabéticos, numéricos e/ou caracteres especiais.
3.17	Exigir que os requisitos de comprimento de senha estabelecidos pela política ou pelo regulamento sejam cumpridos. O uso de oito caracteres é o mais comum, porém usar 16 é mais recomendado. Considere, ainda, o uso de senhas que contenham várias palavras (uma frase).
3.18	A entrada da senha deve ser ocultada na tela do usuário. Em HTML, utilizar o campo dotipo "*password*".
3.19	Desativar a conta após um número pré-definido de tentativas inválidas de autenticação(e.g. cinco tentativas é o mais comum). A conta deve ser desativada por um período suficientemente longo para desencorajar a dedução das credenciais pelo método de força bruta, mas não tão longo ao ponto de permitir um ataque de negação de serviço.
3.20	Os processos de redefinição de senhas e operações de mudanças devem exigir osmesmos níveis de controle previstos para a criação de contas e autenticação.
3.21	Esquemas de pergunta/resposta (pré-definidos) usados para a redefinição da senha devem evitar ataques que lancem respostas aleatórias. Por exemplo, “livro favorito” é uma questão fraca, pois “A Bíblia” é uma resposta muito comum.

3.22	Se optar por usar redefinição de senha baseada em e-mail, envie um e-mail somente para o endereço pré-definido contendo um link ou uma senha de acesso temporário que permitam ao usuário redefinir a senha.
3.23	O tempo de validade das senhas e dos links temporários deve ser curto.
3.24	Exigir a mudança de senhas temporárias na próxima vez que o usuário realizar aautenticação no sistema.
3.25	Notificar o usuário quando a senha for reiniciada (*reset*).
3.26	Prevenir a reutilização de senhas.
3.27	As senhas devem ter, pelo menos, um dia de duração antes de poderem ser alteradas,a fim de evitar ataques de reutilização de senhas.
3.28	Garantir que a troca de senhas está em conformidade com os requisitos estabelecidos na política ou regulamento. Sistemas críticos podem exigir alterações mais frequentes nas credenciais de segurança. O tempo entre as trocas de senhas deve ser controlado administrativamente.
3.29	Desativar a funcionalidade de lembrar a senha nos campos de senha do navegador.
3.30	A data e a hora da última utilização (bem ou malsucedida) de uma conta de usuáriodevem ser comunicadas no próximo acesso ao sistema.
3.31	Realizar monitoramento para identificar ataques contra várias contas de usuários, utilizando a mesma senha. Esse padrão de ataque é utilizado para explorar o uso de senhas padrão.
3.32	Modificar todas as senhas que, por padrão, são definidas pelos fornecedores, bem comoos identificadores de usuários (IDs), ou desativar as contas associadas.
3.33	Exigir nova autenticação dos usuários antes da realização de operações críticas.
3.34	Utilizar autenticação de múltiplos fatores (utilizando simultaneamente token, senha, biometria, etc.) para contas altamente sensíveis ou de alto valor transacional.
3.35	Caso utilize código de terceiros para realizar a autenticação, inspecione-ocuidadosamente para garantir que ele não é afetado por qualquer código

	malicioso.

3.2.4 Gerenciamento de sessões

Praticamente toda sessão de usuário é implementada por meio de um token que identifica a sessão e que é concedido após o usuário se autenticar. A sessão, em si, é um conjunto de estruturas armazenadas no servidor, que mantém controle das interações do usuário com a aplicação.

A grande maioria dos ataques contra o gerenciamento de sessões de uma aplicação busca comprometer o token concedido a outros usuários. Se bem-sucedido, o autor do ataque pode se passar pelo usuário (vítima) como se fosse o usuário autenticado.

Os principais problemas surgem na forma como o token é criado – o que pode permitir a um atacante adivinhar o token de outros usuários - e na forma como os tokens são gerenciados - permitindo que um autor de ataque obtenha o token de outro usuário e, com base no envio de tal token, realize um ataque de personificação. Vide abaixo uma lista de medidas a serem considerados nessa hipótese:

ID	Detalhamento do Controle de Segurança Crítico
4.1	Utilizar controles de gerenciamento de sessão baseados no servidor ou em framework. Aaplicação deve reconhecer apenas esses identificadores de sessão como válidos.
4.2	A criação dos identificadores de sessão deve ser sempre realizada em um sistema confiável - por exemplo, centralizado no servidor.
4.3	O controle de gestão de sessão deve usar algoritmos conhecidos, padronizados e bemtestados que garantam a aleatoriedade dos identificadores de sessão.
4.4	Definir o domínio e o caminho para os cookies que contenham identificadores de sessãoautenticados, para um valor devidamente restrito ao site.

4.5	A funcionalidade de saída (*logout*) deve encerrar completamente a sessão ou conexãoassociada.
4.6	A funcionalidade de saída (*logout*) deve estar disponível em todas as páginas que requerem autenticação.
4.7	Estabelecer um tempo de expiração da sessão que seja o mais curto possível, baseado nobalanceamento dos riscos e requisitos funcionais do negócio. Na maioria dos casos, não deve ser maior que algumas horas.
4.8	Não permitir logins persistentes (sem prazo de expiração), e realizar o encerramento da sessão periodicamente, mesmo quando ela estiver ativa. Isso deve ser feito, especialmente, em aplicações que suportam várias conexões de rede ou que se conectama sistemas críticos. O tempo de encerramento deve estar em sintonia com os requisitos do negócio e o usuário deve receber notificações suficientes para atenuar os impactos negativos dessa medida.
4.9	Se uma sessão estava estabelecida antes do login, ela deve ser encerrada para que umanova seja estabelecida após o login.
4.10	Gerar um novo identificador de sessão quando houver uma nova autenticação.
4.11	Não permitir conexões simultâneas com o mesmo identificador de usuário.
4.12	Não expor os identificadores de sessão em URLs, mensagens de erro ou logs. Os identificadores de sessão devem apenas ser encontrados no cabeçalho do cookie HTTP. Por exemplo, não trafegar os identificadores de sessão sob a forma de parâmetros GET.
4.13	Proteger os dados de sessão do lado servidor contra acessos não autorizados por outros usuários do servidor, através da implementação de controles de acesso apropriados no servidor.

4.14	Gerar um novo identificador de sessão e desativar o antigo periodicamente. Isso pode mitigar certos cenários de ataques de sequestro de sessão (*session hijacking*), quando o identificador de sessão original for comprometido.
4.15	Gerar um novo identificador de sessão caso a segurança da conexão mude de HTTP paraHTTPS, como pode ocorrer durante a autenticação. Internamente à aplicação, é recomendável utilizar HTTPS de forma constante em vez de alternar entre HTTP e HTTPS.
4.16	Utilizar mecanismos complementares ao mecanismo padrão de gerenciamento de sessõespara operações sensíveis do lado servidor, como no caso de operações de gerenciamentode contas, através da utilização de tokens aleatórios ou parâmetros associados à sessão. Esse método pode ser usado para prevenir ataques do tipo *Cross Site Request Forgery* (CSRF).
4.17	Utilizar mecanismos complementares ao gerenciamento de sessões para operações altamente sensíveis ou críticas, utilizando tokens aleatórios ou parâmetros em cada requisição em vez de basear-se apenas na sessão.
4.18	Configurar o atributo "*secure*" para cookies transmitidos através de uma conexão TLS.
4.19	Configurar os cookies com o atributo "*HttpOnly*", a menos que seja explicitamente necessário ler ou definir os valores deles através de scripts do lado cliente da aplicação.

3.2.5 Controle de acesso

Uma das principais etapas tratamento de acesso do usuário é averiguar se cada solicitação individual de acesso deve ser permitida ou negada. Se os mecanismos de validação de identidade estão funcionando corretamente, a aplicação reconhece se as requisições de acesso são válidas.

Uma aplicação pode suportar inúmeras funções de usuários, cada uma envolvendo diferentes combinações de privilégios específicos. Funções específicas podem implementar limites de transação e outras verificações, todas baseadas na identidade do usuário.

Devido à natureza complexa dos requisitos típicos de controle de acesso, tal função é uma

fonte frequente de vulnerabilidades de segurança que permitem a um invasor obter acesso não autorizado a dados e funcionalidades.

Os desenvolvedores costumam fazer suposições erradas sobre como os usuários irão interagir com o aplicativo e frequentemente cometem erros ao dispensar verificações de controle de acesso de algum aplicativo. Analisar essas vulnerabilidades costuma ser trabalhoso, porque essencialmente as mesmas verificações precisam ser repetidas para cada funcionalidade da aplicação.

Devido à prevalência de falhas de controle de acesso, no entanto, esse esforço é uma atividade que pode fornecer benefícios indevidos a um autor de ataque a uma aplicação Web. Vide abaixo uma lista de medidas a serem considerados nessa hipótese.

ID	**Detalhamento do Controle de Segurança Crítico**
5.1	Utilizar apenas objetos do sistema que sejam confiáveis, como ocorre com os objetos desessão do servidor, para realizar a tomada de decisão sobre a autorização de acesso.
5.2	Utilizar um único componente em toda a aplicação Web para realizar o processo de verificação de autorização de acesso. Isto inclui bibliotecas que invocam os serviços externos de autorização.
5.3	Quando ocorrer alguma falha no controle de acesso, ela deve ocorrer de modo seguro,sem que sejam repassados detalhes que possam facilitar ataques direcionados.
5.4	Negar todos os acessos, caso a aplicação não consiga ter acesso às informações contidasna configuração de segurança.
5.5	Garantir o controle de autorização em todas as requisições, inclusive em scripts do ladoservidor, "*includes*" e requisições provenientes de tecnologias do lado cliente.
5.6	Isolar do código da aplicação os trechos de código que contêm lógica privilegiada.

5.7	Restringir o acesso aos arquivos e outros recursos, incluindo aqueles que estão fora docontrole direto da aplicação, somente aos usuários autorizados.
5.8	Restringir o acesso às URLs protegidas somente aos usuários autorizados.
5.9	Restringir o acesso às funções protegidas somente aos usuários autorizados.
5.10	Restringir o acesso às referências diretas aos objetos somente aos usuários autorizados.
5.11	Restringir o acesso aos serviços somente aos usuários autorizados.
5.12	Restringir o acesso aos dados da aplicação somente aos usuários autorizados.
5.13	Restringir o acesso aos atributos e dados dos usuários, bem como informações daspolíticas usadas pelos mecanismos de controle de acesso.
5.14	Restringir o acesso às configurações de segurança relevantes apenas aos usuários autorizados.
5.15	As regras de controle de acesso representadas pela camada de apresentação devemcoincidir com as regras presentes no lado servidor.
5.16	Se o estado dos dados deve ser armazenado no lado cliente, utilizar mecanismos de criptografia e verificação de integridade no lado servidor para detectar possíveis adulterações.
5.17	Garantir que os fluxos lógicos da aplicação obedecem às regras de negócio.
5.18	Limitar o número de transações que um único usuário ou dispositivo pode executar em determinado período de tempo. As transações por período de tempo devem estar acimadas necessidades reais do negócio, mas abaixo o suficiente para impedir ataques automatizados.
5.19	Utilizar o campo "*referer*" do cabeçalho somente como forma de verificação suplementar. Ele não deve ser usado sozinho como forma de validação de autorização, pois pode ter ovalor adulterado.

5.20	Se for permitida a existência de sessões autenticadas por longos períodos, fazer a revalidação periódica da autorização do usuário para garantir que os privilégios não forammodificados e, caso tenham sido, realizar o registro em log do usuário e exigir nova autenticação.
5.21	Implementar a auditoria das contas de usuário e assegurar a desativação de contas não utilizadas. Por exemplo, a conta deve ser desativada não mais do que 30 dias após a expiração da senha.
5.22	A aplicação deve dar suporte à desativação de contas e ao encerramento das sessões quando terminar a autorização do usuário - por exemplo, quando ocorrer alguma alteração dos dados do usuário, situação profissional, processos de negócio etc.
5.23	As contas de serviço ou contas de suporte a conexões provenientes ou destinadas aserviços externos devem possuir o menor privilégio possível.
5.24	Criar uma Política de Controle de Acesso para documentar as regras de negócio da aplicação, tipos de dados e critérios ou processos de autorização, para que os acessos possam ser devidamente concedidos e controlados. Isso inclui identificar requisitos de acessos - tanto para os dados, como para os recursos do sistema.
5.25	Configurar os cabeçalhos HTTP para o uso do C.O.R.S (*Cross-Origin Resource Sharing),* utilizado para permitir ou bloquear o acesso a conteúdo como AJAX, fontes em outros sites, de acordo com as regras de negócio da aplicação.

3.2.6 Criptografia

As aplicações necessitam ser desenhadas com uma arquitetura de criptografia forte para proteger seus dados de acordo com sua classificação. Criptografar todas as informações pode ser inviável, mas não criptografar nenhum dado é um grande risco assumido. Um equilíbrio deve ser buscado, normalmente durante a fase de desenho e projeto da aplicação. Desenhar e desenvolver a arquitetura de criptografia durante o desenvolvimento da aplicação, ou após a aplicação estar pronta, inevitavelmente irá custar muito mais que simplesmente construí-la de forma segura desde o início do desenvolvimento.

Alguns requisitos de alto nível devem ser observados:

- Os módulos de criptografia devem apresentar erros de maneira segura e ser tratadoscorretamente.
- Um gerador de número aleatórios seguro deve ser utilizado.
- O acesso às chaves de criptografia deve ser gerenciado de forma segura.

Ainda sobre criptografia, deve ser observado que o item 6.5, listado na tabela abaixo, deve ser implementado de forma atenta e crítica pelos gestores da área de tecnologia da informação. Isso porque o *Federal Information Processing Standard* (FIPS) 140-3 foi aprovado em 22 de marçode 2019, tendo entrado em vigor em 22 de setembro de 2019[7], em substituição ao FIPS 140-2. Embora os respectivos certificados de validação do FIPS 104-3 ainda não tenham sido emitidos, há a expectativa de que isso ocorra em breve.

ID	Detalhamento do Controle de Segurança Crítico
6.1	Todas as funções de criptografia utilizadas para proteger dados sensíveis dos usuários daaplicação devem ser implantadas em um sistema confiável - neste caso, o servidor.
6.2	A senha mestra deve ser protegida contra acessos não autorizados.
6.3	Quando ocorrer alguma falha nos módulos de criptografia, permitir que ela ocorra demodo seguro.
6.4	Todos os números, nomes de arquivos, GUIDs e strings aleatórias devem ser gerados usando um módulo criptográfico com gerador de números aleatórios, somente se os valores aleatórios gerados forem impossíveis de serem deduzidos.
6.5	Os módulos de criptografia usados pela aplicação devem ser compatíveis com a FIPS 140-2 ou com um padrão equivalente https://csrc.nist.gov/projects/cryptographic-module-validation-program/validated-modules

[7] https://www.nist.gov/news-events/news/2019/05/announcing-approval-and-issuance-fips-140-3-security-requirements

6.6	Estabelecer e utilizar uma política e um processo que defina como é realizado o gerenciamento das chaves criptográficas.

3.2.7 Tratamento de erros e logs

O principal objetivo do tratamento de erros e logs é fornecer informação útil para usuários, administradores e times de resposta a incidentes. Devem-se buscar logs de alta qualidade, com mais sinal do que ruído, de forma a evitar a criação de uma quantidade massiva de logs.

Logs com informação de qualidade normalmente possuem dados sensíveis e devem ser protegidos conforme a Norma Complementar nº 21[9] IN01/DSIC/GSIPR, que trata de diretrizes para o Registro de Eventos, Coleta e Preservação de Evidências de Incidentes de Segurança em Redes nos órgãos e entidades da Administração Pública Federal, direta e indireta. Por geralmente possuírem informações bastante sensíveis, os logs também podem ser um alvo bem atraente para os atacantes.

É importante também garantir que a aplicação apresente erros de forma segura, e que esses erros não vazem informações desnecessariamente.

ID	Detalhamento do Controle de Segurança Crítico
7.1	Não expor informações sensíveis nas repostas de erros, inclusive detalhes de sistema,identificadores de sessão ou informação da conta do usuário.
7.2	Usar mecanismos de tratamento de erros que não mostrem informações de depuração(*debug*) ou informações da pilha de exceção.
7.3	Usar mensagens de erro genéricas e páginas de erro personalizadas.
7.4	A aplicação deve tratar os erros sem se basear nas configurações do servidor.
7.5	A memória alocada deve ser liberada de modo apropriado quando ocorrerem condiçõesde erro.
7.6	O tratamento de erros lógicos associados com os controles de segurança deve, porpadrão, negar o acesso.

7.7	Todos os controles de log devem ser implementados em um sistema confiável, porexemplo, centralizar todo o processo no servidor.
7.8	Os controles de log devem dar suporte tanto para os casos de sucesso como os de falharelacionados com os eventos de segurança.
7.9	Garantir que os logs armazenam eventos importantes.
7.10	Garantir que as entradas de log que incluam dados nos quais não se confia não sejamexecutadas como código-fonte na interface de visualização de logs.
7.11	Restringir o acesso aos logs apenas para pessoal autorizado.
7.12	Utilizar uma rotina centralizada para realizar todas as operações de log.
7.13	Não armazenar informações sensíveis nos registros de logs, como detalhesdesnecessários do sistema, identificadores de sessão e senhas.
7.15	Garantir o uso de algum mecanismo que conduza (ou facilite) o processo de análise delogs.
7.16	Registrar em log todas as falhas de validação de entrada de dados.
7.17	Registrar em log todas as tentativas de autenticação, especialmente as que falharam poralgum motivo.
7.18	Registrar em log todas as falhas de controle de acesso.
7.19	Registrar em log todos os eventos suspeitos de adulteração, inclusive alterações inesperadas no estado dos dados.
7.20	Registrar em log as tentativas de conexão com tokens de sessão inválidos ou expirados.
7.21	Registrar em log todas as exceções lançadas pelo sistema.

7.22	Registrar em log todas as funções administrativas, inclusive as mudanças realizadas nasconfigurações de segurança.
7.23	Registrar em log todas as falhas de conexão TLS com o *backend*.
7.24	Registrar em log todas as falhas que ocorreram nos módulos de criptografia.
7.25	Utilizar uma função de hash criptográfica para validar a integridade dos registros de log.

3.2.8 Proteção de dados

É necessário que a aplicação proteja os dados tratados por ela, de forma que o acesso às suas informações se restrinja ao mínimo necessário. Além disso, é importante adotar controles de segurança ao armazenar as informações para garantir que os dados necessários sejam criptografados e que informações temporárias ou registradas em cache sejam eliminadas quando não forem mais utilizadas.

É preciso também evitar que informações sensíveis sejam transportadas de forma insegura, e evitar que informações sensíveis sobre a aplicação estejam visíveis aos usuários finais.

ID	Detalhamento do Controle de Segurança Crítico
8.1	Implementar uma política de privilégio mínimo, restringindo aos usuários apenas às funcionalidades, dados e informações do sistema que são necessárias para executarem suas tarefas.
8.2	Proteger contra acesso não autorizado todas as cópias temporárias ou registradas em cache que contenham dados sensíveis e estejam armazenadas no servidor; excluir esses arquivos logo que não forem mais necessários.

8.3	Criptografar informações altamente sensíveis quando armazenadas – como dados de verificação de autenticação – mesmo que estejam no lado servidor, usando sempre algoritmos conhecidos, padronizados e bem testados. Consulte a seção que trata sobre “Práticas de Criptografia” para orientações adicionais.
8.4	Proteger o código-fonte presente no servidor para que não seja acessado por usuáriosnão autorizados.
8.5	Não armazenar senhas, strings de conexão ou outras informações confidenciais em textoclaro/legível ou em qualquer forma criptograficamente insegura no lado cliente. Isso é válido também quando há utilização de formatos inseguros, como *MS viewstate* ou códigocompilado que é executado no lado cliente.
8.6	Remover comentários do código de produção que podem ser acessados pelos usuários epodem revelar detalhes internos do sistema ou outras informações sensíveis.
8.7	Remover aplicações desnecessárias e documentação do sistema que possam revelarinformações importantes para os autores de ataques.
8.8	Não incluir informações sensíveis nos parâmetros de requisição HTTP GET.
8.9	Desativar a funcionalidade de autocompletar nos formulários que contenham informaçõessensíveis, inclusive no formulário de autenticação.
8.10	Desativar a cache realizada no lado cliente das páginas que contenham informações sensíveis. O parâmetro "*Cache-Control: no-store*" pode ser usado em conjunto com o controle definido no cabeçalho HTTP “Pragma: no-cache”[8], que é menos efetivo, porém compatível com HTTP/1.0.
8.11	A aplicação deve dar suporte à remoção de dados sensíveis quando estes não forem maisnecessários - como, por exemplo, informação pessoal ou dados financeiros.

[8] https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Headers/Pragma

8.12	Implementar mecanismos de controle de acesso apropriados para dados sensíveis armazenados no servidor. Isto inclui dados em cache, arquivos temporários e dados que devem ser acessíveis somente por usuários específicos do sistema.
8.13	Prover mecanismos que garantam a anonimização [9]dos dados pessoais e dados pessoaissensíveis, conforme a Lei Geral de Proteção de Dados Pessoais brasileira.

3.2.9 Segurança nas comunicações

Para transmissão de dados e informações, é ideal que se utilizem canais de comunicação seguros, o que pode ser implementado utilizando o protocolo TLS ou outra cifra forte. Devem-se utilizar de recomendações mais recentes de boas práticas de configuração para habilitar e ordenar os algoritmos e cifras preferenciais.

Algoritmos e cifras fracos, ou perto de serem considerados obsoletos, devem ser considerados somente em último caso. Algoritmos e cifras conhecidamente inseguros ou obsoletos não devem ser utilizados.

ID	Detalhamento do Controle de Segurança Crítico
9.1	Utilizar criptografia na transmissão de todas as informações sensíveis. Isto deve incluir TLSpara proteger a conexão e deve ser complementado com criptografia de arquivos que contém dados sensíveis ou conexões que não usam o protocolo HTTP.
9.2	Os certificados TLS devem ser válidos, possuir o nome de domínio correto, não estarexpirados e ser instalados com certificados intermediários, quando necessário.
9.3	Quando ocorrer alguma falha nas conexões TLS, o sistema não deve fornecer umaconexão insegura.
9.4	Utilizar conexões TLS para todo o conteúdo que requerer acesso autenticado ou quecontenha informação sensível.

[9] Lei nº 13.709/2018, art. 6º, XI - anonimização: utilização de meios técnicos razoáveis e disponíveis no momento do tratamento, por meio dos quais um dado perde a possibilidade de associação, direta ou indireta, a um indivíduo.

9.5	Utilizar TLS para conexões com sistemas externos que envolvam funções ou informaçõessensíveis.
9.6	Utilizar um padrão único de implementação TLS configurado de modo apropriado.
9.7	Especificar a codificação dos caracteres para todas as conexões.
9.8	Filtrar os parâmetros que contenham informações sensíveis, provenientes do "*HTTPreferer*", nos links para sites externos.

3.2.10 Configuração do sistema

A configuração de uma aplicação recém-lançada deve ser segura o suficiente para estar publicada na Internet, o que significa uma configuração segura por padrão. É importante certificar que a aplicação possua:

- Um ambiente de construção seguro, repetível e automatizado;
- Biblioteca de terceiros reforçada, gerenciamento de dependência e configuração de forma que componentes desatualizados ou inseguros não sejam incluídos na aplicação;
- Uma configuração baseada no conceito *Security by Default* – a qual possui, por padrão,configurações seguras, de forma que a escolha por configurações menos rígidas de segurança seja uma escolha de administradores e usuários.

ID	**Detalhamento do Controle de Segurança** **Crítico**
10.1	Garantir que os servidores, *frameworks* e componentes do sistema estão executando aúltima versão aprovada.
10.2	Garantir que os servidores, *frameworks* e componentes do sistema possuem as atualizações mais recentes e seguras aplicadas para a versão em uso.
10.3	Desativar a listagem de diretórios.

10.4	Restringir, para o mínimo possível, os privilégios do servidor Web, dos processos e dascontas de serviços.
10.5	Quando ocorrerem exceções no sistema, garantir que as falhas ocorram de modoseguro.
10.6	Remover todas as funcionalidades e arquivos desnecessários.
10.7	Remover código de teste ou qualquer funcionalidade desnecessária para o ambiente deprodução, antes de instalar o sistema no servidor de produção.
10.8	Prevenir a divulgação da estrutura de diretórios, impedindo que robôs de busca[10] façam indexação de arquivos sensíveis, através da configuração do arquivo "*robots.txt*"[11]. Os diretórios que não devem ser acessados por estes indexadores devem ser colocados emum diretório isolado. Assim, apenas é necessário negar o acesso ao diretório pai definidono arquivo “robots.txt”, evitando ter que negar o acesso a cada diretório individualmente.
10.9	Definir quais métodos HTTP, GET ou POST a aplicação irá suportar, e se serão tratadosde modo diferenciado nas diversas páginas da aplicação.
10.10	Desativar as extensões HTTP desnecessárias como, por exemplo, o *WebDAV*. Caso seja necessário o uso de alguma extensão HTTP com o propósito de suportar a manipulação de arquivos, utilize um mecanismo de autenticação conhecido, padronizado e bem testado.
10.11	Se o servidor processa tanto requisições HTTP 1.0 como HTTP 1.1, certificar-se de que ambos são configurados de modo semelhante ou assegure que qualquer diferença existente seja compreendida - como, por exemplo, o manuseio de métodos HTTP estendidos.

[10] Robôs são programas de computador que percorrem automaticamente as páginas da Internet em busca de documentos, com o propósito de indexá-los, validá-los ou monitorar alterações de conteúdo. [https://pt.wikipedia.org/wiki/Protocolo_de_exclus%C3%A3o_de_rob%C3%B4s]

[11] Exemplos de como configurar o arquivo robots.txt. [http://www.mundoseo.com.br/seo-tecnico/robotstxt-configuracao-seu-site/]

10.12	Remover informações desnecessárias presentes nos cabeçalhos de resposta HTTP e que podem estar relacionadas com o sistema operacional, versão do servidor Web e frameworks de aplicação.
10.13	O armazenamento da configuração de segurança para a aplicação deve ser capaz de ser produzida de forma legível para dar suporte à auditoria.
10.14	Implementar um sistema de gestão de ativos para manter o registro dos componentes e programas.
10.15	Isolar o ambiente de desenvolvimento da rede de produção e conceder acesso somente para grupos de desenvolvimento e testes. É comum os ambientes de desenvolvimento serem configurados de modo menos seguro do que os ambientes de produção. Deste modo, os autores de ataques podem usar essa diferença para descobrir vulnerabilidades comuns ou encontrar formas de exploração.
10.16	Implementar um sistema de controle de mudanças para gerenciar e registrar as alterações no código, tanto de desenvolvimento, como dos sistemas em produção.
10.17	Rejeitar requisições que utilizem métodos (ou verbos) HTTP além do GET ou POST que não são utilizados pela aplicação (Verb tempering attacks), como, por exemplo, os verbos TRACE e OPTIONS e demais métodos.

3.2.11 Segurança em Banco de Dados

A coleta e o tratamento de dados são tarefas fundamentais para que as organizações consigam, através de análises específicas, traçar metas e alcançar o objetivo final de atendimento aos usuários.

Desta forma, é fundamental que as organizações consigam garantir a proteção dos dados armazenados em banco de dados contra acessos indevidos, ações de hackers e incidentes técnicos.

ID	Detalhamento do Controle de Segurança

	Crítico
11.1	Usar consultas parametrizadas fortemente tipadas.
11.2	Utilizar validação de entrada e codificação de saída e assegurar a abordagem de metacaracteres. Se houver falha, o comando não deverá ser executado no banco de dados.
11.3	Certificar-se de que as variáveis são fortemente tipadas.
11.4	Realizar a codificação (*escaping*) de meta caracteres em instruções SQL[12].
11.5	A aplicação deve usar o menor nível possível de privilégios ao acessar o banco dedados.
11.6	Usar credenciais seguras para acessar o banco de dados.
11.7	Não incluir strings de conexão na aplicação. As strings de conexão devem estar em um arquivo de configuração separado, armazenado em um sistema confiável e as informações devem ser criptografadas.
11.8	Usar procedimentos armazenados (*stored procedures*) para abstrair o acesso aosdados e permitir a remoção de permissões das tabelas no banco de dados.
11.9	Encerrar a conexão assim que possível.
11.10	Remover ou modificar senhas-padrão de contas administrativas. Utilizar senhas robustas (pouco comuns ou difíceis de deduzir) ou implementar autenticação de múltiplos fatores. Desativar qualquer funcionalidade desnecessária no banco de dados, como "*stored procedures*" ou serviços não utilizados. Instalar o conjunto mínimo de componentes ou de opções necessárias (redução da superfície de ataque).
11.11	Eliminar o conteúdo desnecessário incluído por padrão pelo fornecedor como esquemas e bancos de dados de exemplo.

[12] https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

11.12	Desativar todas as contas criadas por padrão e que não sejam necessárias parasuportar os requisitos de negócio.
11.13	A aplicação deve conectar-se ao banco de dados com diferentes credenciais de segurança para cada tipo de necessidade - como, por exemplo, usuário, somente leitura, convidado ou administrador.
11.14	O Banco de Dados deve ser hospedado em um servidor diferente (virtualizados ounão) dos demais serviços.
11.15	Instalar, tão logo quanto possível, *patches* e *hotfixes* de versões mais atuais validadase homologadas.
11.16	Encriptar os Backups de modo a proteger o sigilo das informações em caso de perdaou extravio.

3.2.12 Gerenciamento de Arquivos

É comum que as aplicações recebam arquivos do usuário. Porém, os arquivos fornecidos pelos usuários também são um vetor de ataque bastante utilizados por atacantes. Por meio do envio de arquivos maliciosos, por exemplo, é possível obter novas formas de interagir com o servidor e até executar códigos de forma remota.

Assim, é importante adotar um gerenciamento de arquivos que trate, manipule e armazene de forma segura os arquivos que possam ser manipulados pelos usuários.

ID	**Detalhamento do Controle de Segurança** **Crítico**
12.1	Não repassar dados fornecidos pelos usuários diretamente a uma função de inclusãodinâmica.
12.2	Solicitar autenticação antes de permitir que seja feito o carregamento de arquivos.

12.3	Limitar os tipos de arquivos que podem ser enviados para aceitar somente os necessários ao propósito do negócio.
12.4	Validar se os arquivos enviados são do tipo esperado, através da validação doscabeçalhos, pois realizar a verificação apenas pela extensão é insuficiente.
12.5	Não salvar arquivos no mesmo diretório de contexto da aplicação Web. Os arquivos devem ser armazenados no servidor de conteúdos ou no banco de dados.
12.6	Prevenir ou restringir o carregamento de qualquer arquivo que possa ser interpretadoou executado pelo servidor Web.
12.7	Desativar privilégios de execução nos diretórios de armazenamento de arquivos.
12.8	Implantar o carregamento seguro nos ambientes UNIX por meio da montagem do diretório de destino como uma unidade lógica, usando o caminho associado ou o ambiente de "*chroot*".
12.9	Ao referenciar arquivos, usar uma lista de permissões (*whitelist*) de nomes e de tipos de arquivos permitidos. Realizar a validação do valor do parâmetro passado e, caso ele não corresponda ao que é esperado, rejeitar a entrada ou utilizar um valor padrão.
12.10	Não transmitir, sem nenhum tipo de tratamento, os dados informados pelo usuário a redirecionamentos dinâmicos. Se isso for necessário, o redirecionamento deverá aceitar apenas URLs relativas e validadas.
12.11	Não passar caminhos de diretórios ou de arquivos em requisições. Usar algum mecanismo de mapeamento desses recursos para índices definidos em uma lista pré-definida de caminhos.
12.12	Nunca enviar o caminho absoluto do arquivo para o lado cliente de uma aplicação oupara o usuário.

12.13	Certificar-se de que os arquivos da aplicação e os recursos estão definidos somentecom o atributo de leitura.
12.14	Verificar os arquivos que os usuários submeterem através do mecanismo de carregamento em busca de vírus e *malwares*.
12.15	Limitar o tamanho máximo aceito para *uploads* de arquivos no servidor.

3.2.13 Gerenciamento de memória

Informações sensíveis ou úteis a um atacante podem ser armazenadas ou acessadas na memória por meio de técnicas e funções que permitem esse acesso. Além disso, atacantes podem se utilizar da técnica de transbordamento de dados (*buffer overflow)* -na qual o programa, ao escrever dados em um buffer, ultrapassa o limite de tamanho do buffer, sobrescrevendo a memória adjacente – para acessar endereços de memória ou injetar códigos maliciosos.

Nesse contexto, é de suma importância que os desenvolvedores implementem técnicas e controles que garantam que a aplicação fará o gerenciamento e o uso adequado dos recursos de memória, visando com isso a evitar que processos legítimos sejam impactados por ações internas ou externas que possam comprometer o comportamento esperado do sistema ou degradar a performance da aplicação.

ID	**Detalhamento do Controle de Segurança** **Crítico**
13.1	Utilizar controle de entrada/saída para os dados que não sejam confiáveis.
13.2	Verificar se o *buffer* é tão grande quanto o especificado.
13.3	Ao usar funções que aceitem determinado número de bytes para realizar cópias, como*strncpy*(), estar ciente de que, se o tamanho do *buffer* de destino for igual ao tamanhodo *buffer* de origem, ele não pode encerrar a sequência de caracteres com valor nulo(*null*).

13.4	Verificar os limites do *buffer* caso as chamadas à função sejam realizadas em ciclos e verificar se não há nenhum risco de ocorrer gravação de dados além do espaço reservado.
13.5	Truncar todas as strings de entrada para um tamanho razoável antes de passá-las paraas funções de cópia e concatenação.
13.6	Na liberação de recursos alocados para objetos de conexão, identificadores de arquivo etc., não contar com o "*garbage collector*" e realizar a tarefa explicitamente.
13.7	Usar pilhas não executáveis, quando disponíveis.
13.8	Evitar o uso de funções reconhecidamente vulneráveis, como *printf(), strcat(), strcpy()* etc.
13.9	Liberar a memória alocada de modo apropriado após concluir a sub-rotina (função/método) e em todos os pontos de saída.

3.2.14 Práticas Gerais de Codificação

Com o advento da metodologia de desenvolvimento "*DevOps*", os procedimentos de segurança, adotados outrora pela equipe de arquitetura de segurança, devem ser readequados para que a rotina de análise de segurança de aplicações seja introduzida nesse novo contexto de desenvolvimento ágil.

Os procedimentos de análise de segurança são vistos normalmente como inflexíveis e excessivamente assertivos pelos profissionais de desenvolvimento de aplicações, já que estes podem, em diversas oportunidades, argumentar que existem várias maneiras de resolver um problema. De fato, quando se aborda a arquitetura de softwares e aplicações, não existe solução única e simples para determinado problema.

É provável que uma função específica de uma aplicação Web seja revisada continuamente ao longo de sua vida útil, mas a arquitetura geral raramente mudará e pode passar por evoluções graduais. O mesmo quadro acontece quando se fala de arquitetura de segurança. Ao desenvolver uma aplicação Web com controles de segurança desde sua concepção, a organização interessada provavelmente irá poupar tempo e dinheiro, reduzindo as chances da ocorrência de incidentes de segurança.

ID	Detalhamento do Controle de Segurança Crítico
14.1	Para tarefas comuns, utilizar sempre código testado, gerenciado e aprovado, ao invésde criar código novo e não gerenciado.
14.2	Utilizar APIs que executem tarefas específicas para realizar operações do sistema operacional. Não permitir que a aplicação execute comandos diretamente no sistema operacional, especialmente através da utilização de "*shells*" de comando iniciadas pelaaplicação.
14.3	Utilizar mecanismos de verificação de integridade por "*checksum*" ou "*hash*" para verificar a integridade do código interpretado, bibliotecas, arquivos executáveis e arquivos de configuração.

14.4	Utilizar mecanismos de bloqueio para evitar requisições simultâneas para a aplicação ou utilizar um mecanismo de sincronização para evitar condições de concorrência (*race conditions*).
14.5	Proteger as variáveis compartilhadas e os recursos contra acessos concorrentes inapropriados.
14.6	Instanciar explicitamente todas as variáveis e dados persistentes durante a declaração,ou antes da primeira utilização.
14.7	Quando a aplicação tiver que ser executada com privilégios elevados, aumentar osprivilégios o mais tarde possível e revogá-los logo que seja possível.
14.8	Evitar erros de cálculo decorrentes da falta de entendimento da representação internada linguagem de programação usada e de como é realizada a interação com os aspectos de cálculo numérico.
14.9	Prestar bastante atenção nas discrepâncias de tamanho de byte, precisão, distinções de sinal (*signed/unsigned*), truncamento, conversão e "*casting*" entre os tipos, cálculos que devolvam erros do tipo "*not-a-number*" e, também, como a linguagem de programação trata a representação interna de números muito grandes ou muito pequenos.
14.10	Não transferir diretamente dados fornecidos pelo usuário para qualquer função de execução dinâmica sem antes realizar o tratamento dos dados de modo adequado.
14.11	Restringir a geração e a alteração de código por parte dos usuários.
14.12	Revisar todas as aplicações secundárias, códigos e bibliotecas de terceiros para determinar a necessidade do negócio e validar as funcionalidades de segurança, umavez que estas podem introduzir novas vulnerabilidades.

14.13	Implementar atualizações de modo seguro. Se a aplicação precisar realizar atualizações automáticas, utilizar mecanismos de assinatura digital para garantir a integridade do código e garantir que os clientes façam a verificação da assinatura após descarregarem as atualizações. Usar canais criptografados para transferir o código a partir do host do servidor.

REFERÊNCIAS BIBLIOGRÁFICAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR ISO/IEC 27001:2013**: Tecnologia da informação — Técnicas de segurança — Sistemas de gestão da segurança da informação -Requisitos. Rio de Janeiro, 2013.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR ISO/IEC 27002:2013**: Tecnologia da informação — Técnicas de segurança — Código de prática para controles de segurança dainformação. Rio de Janeiro, 2013.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR ISO/IEC 27005:2019**: Tecnologia da informação — Técnicas de segurança — Gestão de riscos de segurança da informação. Rio deJaneiro, 2019.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR ISO/IEC 27701:2019**: Técnicas de segurança — Extensão da ABNT NBR ISO/IEC 27001 e ABNT NBR ISO/IEC 27002 para gestão da privacidade da informação — Requisitos e diretrizes. Rio de Janeiro, 2019.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR ISO/IEC 31000:2018**: Gestão de Riscos — Diretrizes. Rio de Janeiro, 2018.

BRASIL. Presidência da República. Casa Civil. Subchefia para Assuntos Jurídicos. Lei nº 13.709, de 14 de agosto de 2018. **Lei Geral de Proteção de Dados Pessoais.** Disponível em: < http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm >. Acesso em: 10 abr. 2023.

AUDITSCRITPS. CIS Controls Initial Assessment Tool, versão 7.1d. Disponível em: < https://www.auditscripts.com/download/4229/ >. Acesso: 10 abr. 2023.

BRASIL. Presidência da República. Gabinete de Segurança Institucional. Portaria nº 93, de 26 de setembro de 2019. **Glossário de Segurança da Informação**. Disponível em: < https://www.in.gov.br/en/web/dou/-/portaria-n-93-de-26-de-setembro-de-2019-219115663 >. Acesso em: 10 abr. 2023.

BRASIL. Presidência da República. Gabinete de Segurança Institucional. Departamento de Segurança da Informação e Comunicações. **Instrução Normativa nº 01**, de 27 de maio de

2020. Brasília, DF, GSI/PR, 2020. Disponível em: < https://www.in.gov.br/en/web/dou/-/instrucao-normativa-n-1-de-27-de-maio-de-2020-258915215 >. Acesso em: 10 abr. 2023.

BRASIL. Presidência da República. Gabinete de Segurança Institucional. Departamento de Segurança da Informação e Comunicações. **Instrução Normativa nº 02**, de 24 de julho de 2020. Brasília, DF, GSI/PR, 2020. Disponível em: < https://www.in.gov.br/en/web/dou/-/instrucao-normativa-n-2-de-24-de-julho-de-2020-268684700 > Acesso em 06 abr. 2023

BRASIL. Presidência da República. Gabinete de Segurança Institucional. Departamento de Segurança da Informação e Comunicações. **Norma Complementar nº 16** /IN01/DSIC/GSIPR - Estabelece as Diretrizes para o Desenvolvimento e Obtenção de Software Seguro nos Órgãos e Entidades da Administração Pública Federal, direta e indireta, de 21 de novembro de 2012. Brasília,DF, GSI/PR, 2012. Disponível em: < https://pesquisa.in.gov.br/imprensa/jsp/visualiza/index.jsp?data=21/11/2012&jornal=1&pagina=1&totalArquivos=200 >. Acesso em: 10 abr. 2023.

CENTER INTERNET SECURITY. CIS Controls, versão 8.0. Maio de 2021. Disponível em: https://www.cisecurity.org/controls. Acesso em: 10 abr. 2023.

COMITÊ CENTRAL DE GOVERNANÇA DE DADOS - CCGD. **Guia de Boas Práticas LGPD.** Agosto de 2020. Disponível em: < https://www.gov.br/governodigital/pt-br/seguranca-e-protecao-de-dados/guias/guia_lgpd.pdf >. Acesso em: 10 abr. 2023.

CYBER SECURITY AGENCY OF SINGAPORE (CSA). **Security-by-Design Framework Versão 1.0**.Singapura, 2017. Disponível em: < https://www.csa.gov.sg/docs/default-source/csa/documents/legislation_supplementary_references/security_by_design_framework.pdf?sfvrsn=560b9ff3_0 >. Acesso em: 10 abr. 2023.

INTERNATIONAL STANDARD. **ISO/IEC 29100:2011**: Information technology — Security techniques — Privacy framework. Genebra, 2011.

INTERNATIONAL STANDARD. **ISO/IEC 29134:2017**: Information technology - Securitytechniques – Guidelines for privacy impact assessment. Genebra, 2017.

INTERNATIONAL STANDARD. **ISO/IEC 29151:2017**: Information technology — Security techniques — Code of practice for personally identifiable information protection. Genebra, 2017.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Guia de Aperfeiçoamento da Segurança Cibernética para Infraestrutura Crítica, versão 1.1, 2018. Disponível em: < https://www.uschamber.com/sites/default/files/intl_nist_framework_portugese_finalfull_web.pdf>. Acesso em: 10 abr. 2023.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **Framework for Improving Critical Infrastructure Cybersecurity**, versão 1.1, 2018. Disponível em: < https://doi.org/10.6028/NIST.CSWP.04162018 >. Acesso em: 10 abr. 2023.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **NIST Special Publication 800-53 revisão 5**: Security and Privacy Controls for Information Systems and Organizations. Gaithersburg,2020.

THE OPEN WEB APPLICATION SECURITY PROJECT (OWASP). **Melhores práticas de CodificaçãoSegura – Guia de ReferênciaRápida.** Versão 1.3. Disponível em: <https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/ >. Acesso em: 10 de abr. 2023.

THE OPEN WEB APPLICATION SECURITY PROJECT (OWASP). **Software Assurance Maturity Model.** Versão 2. Disponível em: < https://owasp.org/www-project-samm/ >. Acesso em: 10 de abr. 2023.

ANEXO I

Mudanças desta Versão

Este anexo tem a finalidade de fornecer os destaques das mudanças inseridas nesta versão 2.0 do Guia de Requisitos Mínimos de Privacidade e Segurança da Informação para Aplicações Web em comparação com o documento publicado em abril de 2021.

Primeiramente, ressalta-se que as mudanças inseridas nesta versão em comparação com a anterior visam a adequação com o Guia do Framework de Privacidade e Segurança da Informação v1 elaborado e publicado pela SGD em novembro de 2022.

Foram realizadas adequações no título, de: Guia de Segurança em Aplicações Web, para: Guia de Requisitos Mínimos de Privacidade e Segurança da Informação para Aplicações Web e de: Lei Geral de Proteção de Dados Pessoais (LGPD), para: Programa de Privacidade e Segurança da Informação (PPSI).

Foram feitas inclusões de: seção sobre aviso preliminar e agradecimentos; referência de que controle e medidas do Framework de Privacidade e Segurança da Informação são atendidos pelo Guia de Requisitos Mínimos de Privacidade e Segurança da Informação para Aplicações Web; e ajustes no texto em atualização às práticas recomendadas atualmente.

Foi realizada a inclusão de requisitos de privacidade nas seções: Diretrizes Gerais, Requisitos Gerais e Requisitos Específicos para alinhamento com o Guia do Framework. Além disso, foi realizada a atualização e validação de todas as referências ao longo do documento visando a manutenção do arcabouço referencial.

-