ANEXO XV

RESOLUÇÃO N. 350/2025-TJRO

PODER JUDICIÁRIO DO ESTADO DE RONDÔNIA
SECRETARIA DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO

POLÍTICA DE SEGURANÇA DA INFORMAÇÃO CIBERNÉTICA (PSIC)

NORMA DE SEGURANÇA DA INFORMAÇÃO CIBERNÉTICA

NSIC 13 - DESENVOLVIMENTO E OBTENÇÃO DE SOFTWARE

PRESIDENTE
Desembargador Raduan Miguel Filho

VICE-PRESIDENTE
Desembargador Glodner Luiz Pauletto

CORREGEDOR-GERAL
Desembargador Gilberto Barbosa Batista dos Santos

SECRETÁRIO GERAL
Juiz Rinaldo Forti Silva

SECRETÁRIA DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO

Ângela Carmen Szymczak de Carvalho

COMITÊ GESTOR DE SEGURANÇA DA INFORMAÇÃO MULTIDISCIPLINAR

Desembargador Glodner Luiz Pauletto

Desembargador Gilberto Barbosa Batista dos Santos

Juíza Valdirene Alves da Fonseca Clementele

Elaine Piacentini Bettanin

Jucélio Scheffmacher de Souza

Ângela Carmen Szymczak de Carvalho

Rosemeire Moreira Ferreira

Fabiano Sergio Paiva Dias de Sá

Gustavo Luiz Sevagnan Nicocelli

Eduardo Luiz Will Bezerra

Reginaldo de Souza Gadelha

Ignácio de Loiola Reis Junior

Hilton José de Santana Pinto

COMITÊ DE GESTÃO DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO

Ângela Carmen Szymczak de Carvalho

Alessandra Lima Costa

Reginaldo de Souza Gadelha

Simone Soares Sena de Oliveira

EQUIPE DE ELABORAÇÃO

Allan Tito Leite Ratts

Ângela Carmen Szymczak de Carvalho

Fernanda Soares Lana

Ignacio de Loiola Reis Junior

Jorge Willians da Silva Ferreira Batista
Reginaldo de Souza Gadelha
Sidnei Roberto Feliciano da Silva
Simone Soares Sena de Oliveira
Tárik Kamel de Oliveira
Thiago Fleury Marques Cotrim

REGISTRO DE REVISÕES

Política de Segurança da Informação (PSI)				
Nº	**Data**	**Descrição da Mudança**	**Revisor**	**Aprovador**
1	novembro/2014	Criação da política.	Ignácio	Coinf
2	janeiro/2017	Acréscimo de critérios para acessar o datacenter principal. Formalizada por meio da Resolução n. 036/2016, republicada em 2017 por erro material.	DISEIN DIESE	Tribunal Pleno
3	abril/2019	Atualização da Política. Formalizada por meio da Resolução n. 088/2019.	DISEIN DIESE	CGSI
4	novembro/2020	Alteração na gestão do serviço de correio eletrônico institucional. Formalizada por meio do Ato n. 1.111/2020.	DESEIN DISEIN DIESE	CGSI
5	junho/2021	Atualização sobre a atuação do Comitê Permanente de Segurança do Poder Judiciário. Formalizada por meio da Resolução n. 209/2021.	DESEIN DISEIN DIESE	CGSI
Política da Segurança da Informação Cibernética (PSIC) **NSIC 13 - Desenvolvimento e Obtenção de Software**				
Nº	**Data**	**Descrição da Mudança**	**Revisor**	**Aprovador**
1	junho/2025	Alteração na Resolução para focar na segurança da informação cibernética e criação de anexos específicos para tratar cada tema individualmente.	DESEIN DISEIN DIESE	Cgestic CGSI

1 OBJETIVO

Estabelecer diretrizes, padrões e boas práticas de segurança para o desenvolvimento e obtenção de software seguro no âmbito do Poder Judiciário do Estado de Rondônia.

2 MOTIVAÇÃO

2.1. Disciplinar por meio deste normativo, os requisitos mínimos de segurança da informação e cibernética para o desenvolvimento e obtenção de software seguro no âmbito do PJRO;

2.2 Proteção da Confidencialidade, Integridade, Disponibilidade e Autenticidade das Informações do PJRO;

2.3. Alinhamento com as normas, regulamentações e melhores práticas relacionadas à matéria de segurança da informação e cibernética.

3 FUNDAMENTO LEGAL

3.1 Norma Técnica ABNT NBR ISO/IEC 27001:2022, que especifica os requisitos para estabelecer, implementar, manter e melhorar continuamente um sistema de gestão da segurança da informação dentro do contexto da Organização.

3.2 Norma Técnica ABNT NBR ISO/IEC 27002:2022, que fornece diretrizes para práticas de gestão de segurança da informação.

3.3 Portaria n. 162/2021-CNJ, que aprova Protocolos e Manuais criados pela Resolução CNJ nº 396/2021, que instituiu a Estratégia Nacional de Segurança Cibernética do Poder Judiciário (ENSEC-PJ).

3.4 Lei Geral de Proteção de Dados (LGPD) - Lei nº 13.709/2018, que estabelece regras para o tratamento de dados pessoais, garantindo a privacidade e a proteção das informações, especialmente relevantes no contexto judiciário.

3.5 Lei de Acesso à Informação (LAI) - Lei nº 12.527/2011, que regulamenta o acesso a informações públicas, reforçando a importância da segurança na gestão e divulgação dessas informações.

3.6 National Institute Of Standards And Technology. NIST Special Publication 800-53 revisão 5: Security and Privacy Controls for Information Systems and Organizations.

3.7 CIS Control v8 - Center for Internet Security.

3.8 Melhores Práticas de Codificação Segura OWASP Guia de Referência Rápida. OWASP Secure Coding Practices – Quick Reference Guide.

3.9 The Open Web Application Security Project (OWASP). Software Assurance Maturity Model. Versão 2.

3.10 Guia de Requisitos Mínimos de Privacidade e Segurança da Informação para Aplicações Web. Versão 2 da Secretaria de Governo Digital - SGD.

4 GLOSSÁRIO

4.1 Agente de Ameaça: qualquer entidade que pode causar um impacto negativo num sistema. Pode ser tanto um utilizador mal-intencionado querendo comprometer os controles de segurança do sistema, quanto um desvio acidental do sistema ou uma ameaça física como incêndios ou inundações.

4.2 Autenticação: conjunto de controles usados para verificar a identidade de um utilizador, ou outra entidade que interage com o software.

4.3 Autenticação de Múltiplos Fatores: processo de autenticação que requer vários tipos de credenciais do utilizador. Normalmente é baseado em algo que ele possui (ex.: cartão inteligente), algo que ele sabe (uma ex.: senha), ou em algo que ele é (ex.: dados provenientes de um leitor biométrico).

4.4 Autenticação Sequencial: ocorre quando os dados de autenticação são solicitados em sucessivas páginas, ao invés de serem solicitados numa única página.

4.5 Canonicalização: operação realizada para reduzir várias codificações e representações de dados numa única forma simplificada.

4.6 Caracteres Maliciosos: Quaisquer caracteres ou representações codificadas de caracteres que podem produzir efeitos indesejáveis sobre a operação normal de aplicações ou dos sistemas associados, quando são interpretados por terem significado especial. Estes caracteres podem modificar a estrutura de código ou de declarações, inserir código indesejado, modificar caminhos, causar saídas inesperadas das funções ou rotinas dos programas, causar condições de erro, causar qualquer dos efeitos anteriores em aplicações subjacentes.

4.7 Casos de Uso Impróprio: descrevem o uso de um software de forma prejudicial ou contrário à intenção original do proprietário, como: utilizar para fins ilegais ou antiéticos, explorar falhas de segurança, para fins de assédio ou perseguição, etc.

4.8 Codificação de Entidade HTML: processo de substituição de determinados caracteres ASCII pelas entidades HTML equivalentes. Por exemplo, a codificação poderia substituir o caractere “<” pela entidade HTML equivalente "<". Essas entidades são “inertes” na maioria dos interpretadores – especialmente navegadores – e podem atenuar os ataques do lado do cliente.

4.9 Codificação de Saída Baseada em Contexto: codificação de dados da saída realizada usando como referência o modo como os dados serão utilizados pela aplicação. Se os dados da saída estiverem incluídos na resposta ao cliente, deve-se levar em consideração

situações como: o corpo de um documento HTML, um atributo de HTML, codificação JavaScript, codificação dentro de um CSS ou de uma URL. Devem também ser levados em consideração outros casos como consultas SQL, XML e LDAP.

4.10 Codificação de Saída de Dados: É um conjunto de controles que abordam o uso de codificação para garantir uma saída de dados segura gerada pela aplicação.

4.11 Confidencialidade: princípio que a informação esteja indisponível ou não revelada à pessoa física, ao sistema, ao órgão ou à entidade não autorizada.

4.12 Configuração do Sistema: conjunto de controles que ajudam a garantir que os componentes de infraestrutura de apoio ao software são disponibilizados de forma segura.

4.13 Consultas Parametrizadas (prepared statements): Mantém a consulta e os dados separados através do uso de espaços reservados. A estrutura de consulta é definida por caracteres especiais que representam os parâmetros a serem substituídos. A consulta parametrizada é enviada para a base de dados e preparada para receber os parâmetros e, em seguida, é combinada com os valores dos mesmos. Isto impede que a consulta seja alterada porque os valores dos parâmetros são combinados com a declaração compilada e não concatenados diretamente na sequência de caracteres que compõem a consulta SQL.

4.14 Controle de Acesso: conjunto de controles que liberta ou nega o acesso a um recurso do sistema a um utilizador ou outra entidade qualquer. Normalmente é baseado em regras hierárquicas e privilégios individuais associados a papéis, porém também inclui interações entre sistemas.

4.15 Controles de Segurança: ações que mitigam uma vulnerabilidade potencial e ajudam a garantir que o software se comporte conforme o esperado.

4.16 Cross Site Request Forgery (CSRF): Ocorre quando uma aplicação ou site web externos forçam o navegador do cliente a realizar um pedido involuntário para uma aplicação em que o cliente possui uma sessão ativa. As aplicações são vulneráveis ao CSRF quando usam URLs e parâmetros conhecidos ou previsíveis ou quando o navegador transmite todas as informações da sessão para a aplicação vulnerável de forma automática em cada solicitação.

4.17 Disponibilidade: princípio que a informação esteja acessível e utilizável sob demanda por uma pessoa física ou determinado sistema, órgão ou entidade.

4.18 Estado dos Dados: são dados ou parâmetros usados pela aplicação ou pelo servidor para emular uma ligação persistente ou controlar o estado (status) de um cliente através de um processo de múltiplas pedidos ou transações.

4.19 Exploit: É a ação de aproveitar-se da existência de uma vulnerabilidade. Normalmente é uma ação intencional e tem como objetivo comprometer os controles de segurança do software.

4.20 Gestão de Ficheiros: conjunto de controles que resguardam a interação entre o código da aplicação e os ficheiros do sistema.

4.21 Gestão de Memória: conjunto de controles que dizem respeito ao uso de memória e do buffer.

4.22 Gestão de Sessão: conjunto de controles que tratam da manipulação de sessões HTTP de forma segura por aplicações Web.

4.23 Impacto: É o efeito negativo perceptível para o negócio, resultante da ocorrência de um evento indesejável, que por sua vez pode ser o resultado da exploração de vulnerabilidades.

4.24 Integridade: princípio que a informação não foi modificada ou destruída de maneira não autorizada ou acidental.

4.25 Limites de Confiança: Normalmente, um limite de confiança é constituído pelos componentes do sistema sobre os quais se tem controle direto. Todas as ligações e dados do sistema fora deste controle direto – incluindo todos os clientes e sistemas geridos por terceiros – devem ser considerados como não sendo de confiança e necessitam de validação na fronteira, antes de receberem permissões para realizarem interações com o sistema.

4.26 Mitigar: medidas tomadas para reduzir o grau de severidade de uma vulnerabilidade. Essas medidas incluem a remoção de uma vulnerabilidade, seja ao torná-la mais difícil de ser explorada, ou ao reduzir o impacto negativo de uma exploração bem sucedida.

4.27 Postback: troca de informações entre servidores para reportar as ações de um usuário em um site, canal ou aplicativo.

4.28 Práticas de Criptografia: conjunto de controles usado para garantir que as operações de criptografia dentro da aplicação são executadas de modo seguro.

4.29 Práticas Gerais de Programação: conjunto de controles abrangendo práticas de codificação que não se encaixam facilmente em outras categorias.

4.30 Proteção dos Dados: conjunto de controles para ajudar a garantir que o software trata o armazenamento das informações de modo seguro.

4.31 Requisitos de Segurança: conjunto de requisitos funcionais e de projeto para ajudar a garantir que o software é construído e disponibilizado de forma segura.

4.32 Segurança das Comunicações: conjunto de controles para ajudar a garantir o envio e o recebimento das informações de modo seguro.

4.33 Segurança da Base de Dados: conjunto de controles para garantir que o software interage com as bases de dados de forma segura e que as bases de dados estão configuradas de forma segura.

4.34 Software: qualquer programa, aplicativo ou sistema desenvolvido para utilização em computadores ou em outros dispositivos eletroeletrônicos.

4.35 String (ou cadeia de caracteres): sequência de caracteres utilizada para representar texto. Esses caracteres podem incluir letras, números, símbolos e espaços.

4.36 Tratamento de Erros e Log: conjunto de práticas adotadas para garantir que a aplicação realize o tratamento dos erros de modo seguro e o registo de log dos eventos de modo apropriado.

4.37 Tratamento dos Dados: processo de tornar seguros os dados potencialmente prejudiciais através do processo de remoção, substituição, codificação ou escaping10 dos caracteres.

4.38 Validação de Entrada de Dados: conjunto de controles para verificar se as propriedades de todas as entradas de dados correspondem ao que é esperado pela aplicação, como, por exemplo, tipo dos dados, tamanho, intervalos e conjunto de caracteres aceitáveis que não contenham caracteres maliciosos.

4.39 Vulnerabilidade: fragilidade, fraqueza ou falha em um sistema de informação, rede, processo, ou software que pode ser explorada por ameaças para ganhar acesso não autorizado a recursos ou causar danos.

4.40 Autenticidade: propriedade indicativa de que a informação foi produzida, expedida, modificada ou destruída por uma determinada pessoa física, ou por um determinado sistema, órgão ou entidade.

4.41 *Timestamp* (ou carimbo de data/hora): campo *timestamp*, também conhecido como campo de data e hora, é um tipo de dado usado em banco de dados e software para armazenar o momento preciso que um evento ocorreu. Tem como vantagem a precisão, comparabilidade e automatização.

4.42 Directory Traversal: técnica de ataque que permite a um invasor acessar arquivos e diretórios que estão armazenados fora da pasta raiz do servidor web, explorando vulnerabilidades na validação de entrada de dados para navegar pela estrutura de diretórios do sistema.

4.43 JSON (JavaScript Object Notation): formato de dados leve e de fácil leitura utilizado para troca de informações entre sistemas computacionais.

4.44 Clickjacking (sequestro de clique): técnica maliciosa que induz um usuário a clicar em algo diferente do que o usuário pretende, potencialmente revelando informações confidenciais ou permitindo que outras pessoas assumam o controle de seu computador.

4.45 MIME type (*Multi-purpose Internet Mail Extensions* ou Extensões multifuncionais de correio eletrônico): padrão utilizado para identificar o tipo de conteúdo de um arquivo transmitido pela internet.

4.46 GUIDs (Globally Unique Identifier, ou identificador único universal): sequência alfanumérica gerada por um gerador de números aleatórios criptograficamente seguro (CSPRNG) para garantir sua aleatoriedade e, consequentemente, sua unicidade global em sistemas e aplicações que dependem de identificadores únicos e imprevisíveis, como na geração de tokens e outros elementos de segurança.

4.47 NSIC: Norma de Segurança da Informação Cibernética.

5 CONTROLES

Seção I

Do Processo de Desenvolvimento Interno

5.1 Faz parte das etapas de desenvolvimento de software:

5.1.1 Identificar os responsáveis pela definição e validação dos requisitos de segurança que o software deva atender;

5.1.2 Definir os requisitos de segurança logo no início de qualquer projeto de desenvolvimento;

5.1.3 Definir/implementar os controles de segurança necessários para proteger os ativos de informação, de acordo com a sua criticidade;

5.1.4 Implementar controles de segurança por múltiplas camadas, de acordo com a criticidade das informações tratadas pelo software;

5.1.5 Definir e implementar políticas de codificação segura, incluindo diretrizes para a escrita de código limpo e livre de vulnerabilidades conhecidas;

5.1.6 Incorporar testes de segurança contínuos no processo de desenvolvimento, incluindo testes automáticos e manuais, para identificar e corrigir vulnerabilidades de segurança em todas as fases do desenvolvimento;

5.1.7 Manter estrita separação entre os ambientes de desenvolvimento, teste e produção para evitar a propagação de vulnerabilidades e a contaminação de dados;

5.1.8 Adotar uma abordagem DevSecOps para integrar a segurança de forma mais eficaz e eficiente no processo de desenvolvimento ágil;

5.1.9 Estabelecer definições sobre a custódia de código-fonte e manutenção do software.

5.2 Fará parte da gestão de riscos durante o ciclo de desenvolvimento e manutenção de software:

5.2.1 Realizar análises de risco detalhadas durante o ciclo de vida do desenvolvimento para identificar vulnerabilidades potenciais e determinar as medidas de mitigação apropriadas;

5.2.2 Estabelecer e manter um controle de versão rigoroso e com revisões de código para rotinas de codificação de entrada e saída, para garantir que alterações sejam rastreadas e auditadas;

5.2.3 Utilizar ferramentas de segurança automatizadas, como análise estática de código e scanners de vulnerabilidades, para ajudar na detecção precoce de problemas de segurança;

5.2.4 Implementar revisões de código e auditorias regulares por parte de especialistas em segurança, para garantir a conformidade com as políticas e diretrizes de segurança;

5.2.5 Implementar monitoramento contínuo e registros detalhados das atividades no sistema para facilitar a detecção e investigação de atividades suspeitas ou maliciosas;

5.2.6 Desenvolver e manter um plano de resposta a incidentes de segurança, que defina como identificar, reportar e responder a violações de segurança no software.

5.3 Para garantir a proteção a propriedade intelectual de softwares desenvolvidos pelo PJRO, segurança e integridade dos sistemas:

5.3.1 Estabelecer acordos de licenciamento, propriedade dos códigos e direitos de propriedade intelectual condizentes com o interesse deste PJRO;

5.3.2 Estabelecer processos para o gerenciamento seguro de bibliotecas e componentes de terceiros, incluindo a verificação de vulnerabilidades e a manutenção de um inventário de todas as dependências;

5.3.3 Seguir as recomendações e orientações do OWASP para desenvolvimento seguro;

5.3.4 Seguir as políticas robustas de senhas e autenticação, conforme previsto na "NSIC 01 - Gestão de Identidade e Controle de Acesso aos Recursos de TIC" desta Política.

5.3.5 Estabelecer uma política de segurança de dados de entrada e saída que define como os dados devem ser tratados, codificados, e comunicados;

5.3.6 Garantir que o desenvolvimento de software esteja em conformidade com as leis e regulamentos nacionais e internacionais aplicáveis, incluindo normas de proteção de dados e privacidade;

5.3.7 Revisar e atualizar regularmente os requisitos de segurança para se proteger das ameaças de segurança emergentes e atender as necessidades do PJRO;

5.3.8 Fornecer treinamento regular em segurança para os desenvolvedores e outras partes interessadas para manter a equipe atualizada sobre as melhores práticas de segurança e consciente das ameaças atuais;

Seção II

Dos Softwares e Soluções de Terceiros

5.4 A obtenção de softwares e soluções de terceiros deverá observar as seguintes regras:

5.4.1 Definir os requisitos de segurança logo no início de qualquer projeto de obtenção de software, seja por contratação ou doação;

5.4.2 Definir e documentar os requisitos específicos de segurança para a aplicações a serem desenvolvidas externamente;

5.4.3 Submeter o software/solução ao Processo para Homologação de Softwares de terceiros;

5.4.4 Realizar avaliação detalhada da segurança dos fornecedores, incluindo suas práticas de desenvolvimento de software, políticas de segurança da informação e histórico de incidentes de segurança;

5.4.5 Definir claramente as responsabilidades de segurança contínuas do fornecedor no termo de referência, contrato a ser firmado e logo após a implementação do software;

5.4.6 Analisar as licenças de software para garantir compatibilidade com as necessidades e políticas do PJRO, incluindo questões de propriedade intelectual e de conformidade legal;

5.4.7 Estabelecer SLAs específicos relacionados à segurança com fornecedores, incluindo tempos de resposta para atualizações de segurança, patches e suporte em caso de incidentes;

5.4.8 Assegurar que o software de terceiros possa ser integrado de maneira segura com os sistemas existentes, sem introduzir vulnerabilidades;

5.4.9 Providenciar treinamento adequado para os usuários e administradores do software de terceiros, focando nas práticas de uso seguro e conscientização sobre segurança.

Seção III

Da Validação dos Dados de Entrada

5.5 A validação dos dados de entrada deverá atender os seguintes requisitos:

5.5.1 Centralizar e manter uma rotina de validação de dados padronizada para promover a reutilização e a consistência na aplicação, e criar uma camada de confiança única e controlada;

5.5.2 Identificar e classificar todas as fontes de dados de acordo com seu nível de confiança e validar rigorosamente os dados de fontes não confiáveis (ex: entradas de usuário, bases de dados externas);

5.5.3 Especificar um conjunto de caracteres padrão, como UTF-8, para todas as entradas e garantir que a aplicação possa lidar adequadamente com diferentes codificações;

5.5.4 Antes da validação, converter os dados para um formato canônico comum, o que ajuda na prevenção de ataques de injeção e outros vetores de ataque relacionados a codificação;

5.5.5 Ao encontrar falhas na validação, a aplicação deve rejeitar os dados e, se apropriado, registrar o evento e alertar os administradores;

5.5.6 Se o sistema suportar conjuntos de caracteres UTF-8 estendidos, realizar a validação após a correta decodificação dos caracteres;

5.5.7 Validar rigorosamente todos os dados recebidos de clientes, incluindo parâmetros de URL, campos de formulários e cabeçalhos HTTP, utilizando mecanismos de postback automáticos quando necessário;

5.5.8 Garantir que os cabeçalhos HTTP das requisições e respostas contenham apenas caracteres ASCII para prevenir injeções e outras vulnerabilidades;

5.5.9 Verificar os dados recebidos através de redirecionamentos, uma vez que estes podem ser manipulados para introduzir dados maliciosos;

5.5.10 Confirmar que os dados de entrada correspondem aos tipos e intervalos esperados;

5.5.11 Assegurar que o tamanho dos dados de entrada esteja dentro dos limites estabelecidos;

5.5.12 Onde aplicável, utilizar listas brancas para definir explicitamente os caracteres e formatos permitidos nas entradas de dados;

5.5.13 Implementar medidas de segurança adicionais, como a codificação de saída e o uso de APIs seguras, se caracteres potencialmente perigosos forem necessários nas entradas;

5.5.14 Incluir verificações para padrões de entrada específicos que possam ser explorados, como bytes nulos e sequências de caracteres que manipulam caminhos de acesso;

5.5.15 Expandir a rotina de validação padrão para abordar entradas específicas, incluindo mas não limitado a:

5.5.15.1 Verificar e rejeitar entradas que contenham bytes nulos, os quais podem ser usados para terminar prematuramente uma string em algumas linguagens de programação ou sistemas;

5.5.15.2 Inspecionar entradas para identificar e eliminar caracteres de controle como retornos e quebras de linha, que podem afetar o fluxo de dados ou ser utilizados em ataques de injeção;

5.5.15.3 Proteger contra Directory Traversal e ataques de manipulação de caminho, verificando a presença de sequências como "../" ou "..", e representações alternativas em codificações de caracteres estendidos;

5.5.15.4 Utilizar a canonicalização para resolver problemas de codificação dupla ou outras formas de ofuscação que os atacantes possam usar para esconder entradas maliciosas;

5.5.15.5 Inspetar entradas para comandos ou sequências que possam ser interpretados pelo sistema operacional ou pelo banco de dados como instruções executáveis;

5.5.15.6 Verificar entradas para evitar Cross-site Scripting (XSS), assegurando que os dados inseridos não possam ser interpretados como script;

5.5.15.7 Examinar entradas para prevenir SQL Injection, assegurando que caracteres especiais e sequências de escape sejam tratados de forma apropriada;

5.5.15.8 Caso a aplicação utilize XML ou JSON, validar contra ataques como XML External Entities (XXE) e Insecure Deserialization;

5.5.16 Além de validar, é importante sanitizar os dados de entrada, removendo ou substituindo caracteres não seguros;

5.5.17 Utilizar frameworks e bibliotecas de validação de dados comprovados, que oferecem rotinas de validação e sanitização de dados já testadas e seguras;

5.5.18 Realizar validação tanto no lado do cliente quanto no servidor para segurança, pois a validação no cliente pode ser contornada.

5.5.19 Implementar limitações no número de tentativas de envio de formulários para prevenir ataques de força bruta e automatizados;

5.5.20 Para formulários web e pontos de entrada críticos, considerar a implementação de CAPTCHAs para diferenciar entre usuários humanos e bots automatizados;

5.5.21 Manter registros detalhados das validações de dados para permitir auditorias e ajudar na investigação de incidentes de segurança;

5.5.22 Validar dados com base no contexto em que serão usados. Por exemplo, dados que serão inseridos em um banco de dados devem ser validados de maneira diferente de dados que serão exibidos em uma página web;

5.5.23 Assegurar que as mensagens de erro resultantes da validação de dados não exponham detalhes sensíveis da aplicação ou do sistema;

5.5.24 Se a aplicação permite o upload de arquivos, implementar validações específicas para o tipo de arquivo, tamanho, e verificar se não há conteúdo malicioso;

5.5.25 Assegurar que a aplicação possa validar e manipular corretamente dados em diferentes idiomas e conjuntos de caracteres;

5.5.26 Revisar e atualizar periodicamente as regras de validação para se adaptar a novas ameaças e mudanças nos padrões de entrada dos usuários;

5.5.27 Definir políticas claras sobre a retenção de dados e como os dados de entrada deverão ser armazenados ou arquivados.

Seção IV

Da Codificação de Dados de Saída

5.6 A codificação de dados de saída deverá atender os seguintes requisitos:

5.6.1 Centralizar a codificação de dados no servidor para criar um sistema de confiança que gerencie todas as transformações de dados de saída.

5.6.2 Utilizar rotinas padrão, consistentes e testadas para a codificação de dados de saída, garantindo a manutenção da qualidade e segurança;

5.6.3 Implementar codificação baseada no contexto de uso dos dados, utilizando métodos apropriados para a codificação de saída, como entidades HTML, codificação de URLs, e escapamento de caracteres especiais para SQL, XML e LDAP, entre outros;

5.6.4 Codificar todos os dados de saída, a menos que tenha sido comprovado que são seguros para o interpretador de destino, para prevenir ataques como Cross-site Scripting (XSS);

5.6.5 Aplicar limpeza (sanitização) de dados, com base em contexto, para todos os dados provenientes de fontes não confiáveis, especialmente quando utilizados para construir consultas SQL, XML, LDAP e outros tipos de consultas a sistemas de back-end;

5.6.6 Escapar ou remover caracteres especiais de todos os dados que serão utilizados em comandos para o sistema operacional, para prevenir ataques de injeção de comandos;

5.6.7 Validar os dados antes de enviá-los ao usuário, para garantir que estão corretos e são apropriados;

5.6.8 Garantir a segurança dos dados de saída em trânsito, utilizando protocolos criptografados como TLS/SSL para prevenir a interceptação e manipulação dos dados;

5.6.9 Incluir cabeçalhos de segurança HTTPS apropriados para prevenir ataques baseados em resposta, como clickjacking e sniffing (detecção) de MIME type;

5.6.10 Monitorar e registrar a saída de dados sensíveis para detectar possíveis vazamentos de informação e comportamentos anômalos;

5.6.11 Promover o treinamento dos desenvolvedores nas melhores práticas de codificação de saída;

5.6.12 Utilizar listas de permissões para definir explicitamente quais dados são aceitáveis para saída, garantindo que apenas conteúdo seguro seja enviado aos clientes;

5.6.13 Assegurar que os MIME Types dos dados de saída sejam corretamente definidos e validados para prevenir a execução de tipos de arquivo não intencionais;

5.6.14 Implementar cabeçalhos HTTPS para controlar o cache de dados sensíveis e prevenir que informações sejam armazenadas de forma insegura em caches locais;

5.6.15 Realizar testes para garantir que a integridade dos dados não seja comprometida durante a transformação e codificação;

5.6.16 Assegurar que os logs de saída sejam gerenciados de forma segura, protegendo-os contra acesso não autorizado e manipulação;

5.6.17 Ao construir saídas, especialmente em linguagens de script ou páginas HTML, é crucial separar claramente os dados de controle (como scripts e comandos) dos dados fornecidos pelo usuário;

5.6.18 Utilizar *templates* que automaticamente realizam a codificação de saída adequada para prevenir a inserção de conteúdo inseguro;

5.6.19 Definir políticas claras sobre a retenção de dados e como os dados de saída deverão ser armazenados ou arquivados.

Seção V
Autenticação e Gestão de Credenciais

5.7 A autenticação e gestão de credenciais deverá atender os seguintes requisitos:

5.7.1 Exigir autenticação para todas as páginas e recursos que não sejam explicitamente públicos.

5.7.2 Os mecanismos de autenticação devem operar em um sistema seguro e confiável, idealmente centralizando o processo no servidor.

5.7.3 Utilizar serviços de autenticação padronizados e comprovadamente seguros.

5.7.4 Utilizar uma implementação centralizada para gerenciar procedimentos de autenticação.

5.7.5 Isolar a lógica de autenticação da lógica de negócios e usar redirecionadores para controladores de autenticação centralizados.

5.7.6 Definir e executar procedimentos de fallback seguros em casos de falhas nos controladores de autenticação.

5.7.7 Garantir que as funções administrativas sejam tão seguras quanto o mecanismo de autenticação mais sensível da aplicação.

5.7.8 Armazenar senhas usando hashes criptográficos com sal (salted hashes) e evitar o uso de algoritmos fracos ou obsoletos.

5.7.9 Realizar a geração de hashes de senhas em um ambiente controlado e seguro.

5.7.10 Realizar a validação dos dados de autenticação somente após todas as entradas de dados serem processadas.

5.7.11 Usar mensagens de erro genéricas que não revelem qual parte da autenticação falhou.

5.7.12 Utilizar autenticação para ligação a sistemas externos que envolvam tráfego de informação sensível ou acesso a funções.

5.7.13 As credenciais de autenticação para aceder a serviços externos à aplicação devem ser cifradas e armazenadas num local protegido num sistema de confiança, como por exemplo, no servidor da aplicação.

5.7.13.1 É vedado usar o código-fonte para armazenar as credenciais de autenticação.

5.7.14 Utilizar apenas pedidos POST para transmitir credenciais de autenticação;

5.7.15 Somente trocar senhas (não temporárias) através de uma ligação protegida (SSL/TLS) ou como dado cifrado, como no caso de envio de e-mail cifrado. Senhas temporárias enviadas por e-mail podem ser um caso de exceção;

5.7.16 Os requisitos de complexidade de senha estabelecidos pela política de segurança da informação ou regulamento devem ser cumpridos. As credenciais de autenticação devem ser suficientes para resistir a ataques que, tipicamente, ameaçam o ambiente de produção;

5.7.17 A entrada da senha deve ser ocultada no dispositivo de apresentação do utilizador. Em HTML, utilize o campo com o tipo "password";

5.7.18 Desativar a conta após um número pré-definido de tentativas inválidas de autenticação. A conta deve ser desativada por um período de tempo 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;

5.7.19 Os processos de redefinição de senhas e operações de mudanças devem exigir os mesmos níveis de controle previstos para a criação de contas e autenticação;

5.7.20 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 senha de acesso temporário que permitam ao utilizador redefinir a senha;

5.7.21 O tempo de validade das senhas e dos links temporários deve ser curto, no máximo de 1 (uma) hora;

5.7.22 Exigir a mudança de senhas temporárias na próxima vez que o utilizador realizar a autenticação no sistema;

5.7.23 Notificar o utilizador quando a sua senha for reiniciada (reset);

5.7.24 Prevenir a reutilização de senhas;

5.7.25 As senhas devem ter, pelo menos, um dia de duração antes de poderem ser alteradas, a fim de evitar ataques de reutilização de senhas;

5.7.26 Garantir que a troca de senhas está em conformidade com os requisitos estabelecidos na política de segurança da informação ou regulamento. Sistemas críticos podem exigir alterações mais frequentes nas credenciais de segurança. O tempo entre as trocas de senhas deve ser controlado administrativamente;

5.7.27 Desativar a funcionalidade de lembrar a senha nos campos de senha do navegador;

5.7.28 A data/hora da última utilização (bem ou mal sucedida) de uma conta de utilizador deve ser comunicada na próxima entrada no sistema;

5.7.29 Modificar todas as senhas que, por padrão, são definidas pelos fornecedores, bem como os identificadores de utilizadores (IDs) ou desativar as contas associadas;

5.7.30 Exigir nova autenticação dos utilizadores antes da realização de operações críticas;

5.7.31 Utilizar autenticação de múltiplos fatores (utilizando simultaneamente token, senha, biometria, e outras);

5.7.32 Caso utilize código de terceiros para realizar a autenticação, inspecione-o cuidadosamente para garantir que o mesmo não é afetado por qualquer código malicioso;

5.7.33 Manter um registro detalhado de todas as tentativas de acesso, bem-sucedidas e fracassadas, para análise e detecção de atividades suspeitas;

5.7.34 Implementar um gerenciamento de sessão seguro, que inclua timeouts de sessão e invalidação de tokens após o logout.

Seção VI
Gestão de Sessões

5.8 A gestão de sessões deverá atender os seguintes requisitos:

5.8.1 Utilizar controles de gestão de sessão que sejam baseados no servidor ou fornecidos por um framework confiável, reconhecendo apenas os identificadores de sessão gerados por esses métodos;

5.8.2 Assegurar que a criação de identificadores de sessão seja realizada em um ambiente seguro, idealmente no servidor;

5.8.3 Empregar algoritmos robustos e testados para a geração de identificadores de sessão que assegurem sua aleatoriedade e imprevisibilidade;

5.8.4 Restringir o domínio e o caminho nos cookies de sessão para limitar seu uso apenas ao site apropriado;

5.8.5 Garantir que a funcionalidade de logout termine a sessão de maneira eficaz e segura;

5.8.6 Disponibilizar a opção de logout em todas as páginas autenticadas para que os usuários possam encerrar suas sessões facilmente;

5.8.7 Estabelecer um tempo de expiração de sessão que reflita um equilíbrio entre segurança e usabilidade. As sessões devem expirar após um período de inatividade ou após um tempo fixo;

5.8.8 Evitar sessões permanentes e forçar a renovação da autenticação periodicamente para manter a segurança;

5.8.9 Destruir qualquer sessão existente e criar uma nova sessão após o login bem-sucedido para prevenir a fixação de sessão;

5.8.10 Gerar um novo identificador de sessão a cada autenticação para mitigar o risco de sequestro de sessão;

5.8.11 Não permitir múltiplas sessões simultâneas para o mesmo usuário;

5.8.12 Evitar a exposição de identificadores de sessão em locais inseguros como URLs, mensagens de erro ou logs;

5.8.13 Proteger os dados de sessão no servidor contra acesso não autorizado;

5.8.14 Renovar periodicamente o identificador de sessão, especialmente após uma alteração de estado significativa na sessão;

5.8.15 Manter o uso consistente de HTTPS, em vez de alternar entre HTTP e HTTPS;

5.8.16 Empregar tokens de segurança ou parâmetros adicionais para operações sensíveis, ajudando a prevenir ataques como CSRF;

5.8.17 Utilizar tokens únicos para cada requisição em operações altamente sensíveis, em vez de depender unicamente da sessão;

5.8.18 Configurar cookies para serem transmitidos apenas em conexões seguras, definindo o atributo "Secure";

5.8.19 Definir o atributo “HttpOnly” nos cookies para impedir o acesso via scripts do lado do cliente, a menos que haja uma necessidade específica;

5.8.20 Além da autenticação, gerar novos identificadores de sessão após qualquer alteração significativa no nível de acesso do usuário (por exemplo, ao mudar de um usuário comum para um perfil administrativo);

5.8.21 Verificar a validade da sessão em cada requisição para garantir que ela não foi invalidada ou expirada;

5.8.22 Armazenar dados de sessão de forma segura usando mecanismos de armazenamento do lado do servidor que protejam contra manipulação e leitura não autorizada;

5.8.23 Implementar um limite de tempo para inatividade, encerrando a sessão após um período sem atividade do usuário, e não apenas baseado na duração total da sessão;

5.8.24 Garantir que o aplicativo não aceite identificadores de sessão definidos previamente pelo navegador antes do login para prevenir ataques de fixação de sessão;

5.8.25 Utilizar tokens de sincronização para prevenir ataques de requisição de sites cruzados (CSRF) em conjunto com outros mecanismos de prevenção;

5.8.26 Manter registros detalhados das atividades de sessão para auditorias, análise forense e detecção de comportamentos anômalos;

5.8.27 Implementar um sistema de alerta para notificar os usuários sobre atividades suspeitas relacionadas às suas sessões, como logins de locais incomuns ou múltiplas falhas de autenticação;

5.8.28 Para aplicações que permitem, considerar o uso de tokens de autenticação sem estado, como JWT (JSON Web Tokens), que podem ser validados independentemente sem a necessidade de armazenamento de estado do lado do servidor;

5.8.29 Se os valores de sessão puderem ser manipulados via JavaScript, implementar controles rigorosos para prevenir a manipulação de sessão via scripts do lado do cliente;

5.8.30 Desenvolver uma política de sessão que seja personalizada para as necessidades específicas do aplicativo e do ambiente operacional.

Seção VII
Controle de Acesso

5.9 O controle de acesso deverá atender os seguintes requisitos:

5.9.1 Utilizar objetos de sessão do servidor, que são considerados de confiança, para tomar decisões de autorização de acesso.

5.9.2 Implementar um componente único em toda a aplicação para gerenciar a autorização, incluindo chamadas para serviços externos.

5.9.3 Assegurar que qualquer falha no controle de acesso resulte em negação de acesso por padrão.

5.9.4 Caso a aplicação não consiga acessar as configurações de segurança, o acesso deve ser negado.

5.9.5 Exigir controle de autorização para todos os pedidos, incluindo aqueles feitos por tecnologias do lado cliente.

5.9.6 Separar claramente e isolar o código que contém lógica de controle de acesso privilegiada.

5.9.7 Limitar o acesso a arquivos e outros recursos a usuários devidamente autorizados.

5.9.8 As regras de controle de acesso representadas pela camada de apresentação devem coincidir com as regras presentes no lado servidor;

5.9.9 Se o estado dos dados deve ser armazenado no lado do cliente, utilizar mecanismos de criptografia e verificação de integridade no lado servidor para deletar possíveis adulterações;

5.9.10 Garantir que os fluxos lógicos da aplicação obedecem as regras de negócio;

5.9.11 limitar o número de transações que um único utilizador ou dispositivo pode executar em determinado período de tempo. As transações por período de tempo devem estar acima das necessidades reais do negócio, mas abaixo o suficiente para impedir ataques automatizados;

5.9.12 Utilizar o campo “referer” do cabeçalho somente como forma de verificação suplementar. O mesmo não deve ser usado sozinho como forma de validação de autorização porque ele pode ter o valor adulterado;

5.9.13 Se for permitida a existência de sessões autenticadas por longos períodos de tempo, fazer a revalidação periódica da autorização do utilizador para garantir que os privilégios

não foram modificados e, caso tenham sido, realizar o registo em log do utilizador e exigir nova autenticação;

5.9.14 Implementar a auditoria das contas de utilizador 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;

5.9.15 A aplicação deve dar suporte à desativação de contas e ao encerramento das sessões quando terminar a autorização do utilizador. Por exemplo: quando ocorrer alguma alteração dos dados do utilizador, situação profissional, processos de negócio etc.;

5.9.16 As contas de serviço ou contas de suporte a ligações provenientes ou destinadas a serviços externos devem possuir o menor privilégio possível;

5.9.17 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. Isto inclui identificar requisitos de acessos, tanto para os dados, como para os recursos do sistema;

5.9.18 Assegurar que cada usuário tenha apenas os privilégios estritamente necessários para realizar suas tarefas;

5.9.19 Implementar controles para garantir que as funções críticas do sistema sejam separadas e que nenhuma única conta tenha controle total sem supervisão;

5.9.20 Confirmar que as regras de controle de acesso sejam consistentemente aplicadas em todas as camadas da aplicação;

5.9.21 Permitir que as permissões sejam alteradas dinamicamente e que o sistema reflita essas mudanças em tempo real sem necessidade de reiniciar ou recarregar a configuração;

5.9.22 Implementar controles de acesso que levem em consideração o nível de risco associado a diferentes tipos de dados e operações do usuário;

5.9.23 Permitir auditorias regulares de acessos concedidos para garantir que eles ainda são necessários e adequados.

5.9.24 Deve ser restringindo somente aos utilizadores autorizados, os seguintes acessos:

5.9.24.1 URLs protegidas.

5.9.24.2 Funções protegidas.

5.9.24.3 Às referências diretas aos objetos.

5.9.24.4 Aos serviços;

5.9.24.5 Aos dados da aplicação.

5.9.24.6 Aos atributos e dados dos utilizadores, bem como informações das políticas usadas pelos mecanismos de controle de acesso.

5.9.24.7 Às configurações de segurança relevantes apenas aos utilizadores autorizados.

Seção VIII

Práticas de Criptografia

5.10 As práticas de criptografia deverá atender os seguintes requisitos:

5.10.1 Assegurar que todas as funções de criptografia usadas para proteger dados sensíveis sejam implementadas em um sistema seguro e confiável, como o servidor da aplicação;

5.10.2 Salvaguardar chaves mestras e outras chaves criptográficas importantes contra acesso não autorizado, usando cofres de chaves (key vaults) ou módulos de segurança de hardware (HSMs);

5.10.3 Garantir que qualquer falha nos módulos de criptografia resulte em um estado seguro que preserve a confidencialidade dos dados;

5.10.4 Utilizar geradores de números aleatórios criptograficamente seguros (CSPRNG) para todas as operações que dependem de aleatoriedade, como a geração de tokens, *strings* aleatórias e *GUIDs*;

5.10.5 Assegurar que os módulos de criptografia utilizados atendam a padrões reconhecidos, como FIPS 140-2 ou equivalentes, para a proteção de informações sensíveis;

5.10.6 Estabelecer e manter uma política e procedimento robustos para a gestão das chaves criptográficas, incluindo geração, armazenamento, rotação e revogação de chaves;

5.10.7 Criptografar dados sensíveis em trânsito usando TLS/SSL e em repouso, por exemplo, em bancos de dados ou armazenamento de arquivos;

5.10.8 Implementar uma estratégia de rotação de chaves para substituir as chaves criptográficas periodicamente ou em resposta a eventos de segurança;

5.10.9 Utilizar algoritmos de criptografia modernos, seguros e sem vulnerabilidades conhecidas. Evitar o uso de algoritmos considerados fracos ou obsoletos;

5.10.10 Escolher um tamanho de chave apropriado para a criptografia, baseando-se na sensibilidade dos dados e nas melhores práticas atuais;

5.10.11 Utilizar mecanismos seguros de armazenamento de chaves, evitando armazenamento no código-fonte ou em locais facilmente acessíveis;

5.10.12 Empregar envelopes de chaves quando for necessário transmitir chaves criptográficas de forma segura;

5.10.13 Estabelecer uma política para a rotação periódica de chaves criptográficas, reduzindo o risco caso uma chave seja comprometida;

5.10.14 Quando possível, implementar criptografia de ponta a ponta para proteger os dados durante todo o seu percurso, desde o cliente até o servidor;

5.10.15 Realizar testes regulares para identificar e corrigir vulnerabilidades nos componentes criptográficos, incluindo a configuração de protocolos, a implementação de algoritmos e a gestão de chaves;

5.10.16 Armazenar chaves de criptografia separadamente dos dados que elas protegem, idealmente em sistemas ou locais diferentes;

5.10.17 Manter registros detalhados das atividades criptográficas, incluindo acesso a chaves, geração de chaves, uso de chaves e alterações nas configurações de criptografia;

5.10.18 Providenciar treinamento regular para os desenvolvedores e operadores de sistemas sobre as práticas de criptografia, incluindo o uso adequado de algoritmos, gestão de chaves e atualizações de segurança.

Seção IX

Tratamento de Erros

5.11 O tratamento de erros deverá atender os seguintes requisitos:

5.11.1 Evitar a exposição de informações sensíveis, como detalhes do sistema, identificadores de sessão ou informações de contas de usuários, em mensagens de erro;

5.11.2 Implementar mecanismos de tratamento de erros que não exibam informações de depuração ou detalhes da pilha de exceções ao usuário final;

5.11.3 Utilizar mensagens de erro genéricas e páginas de erro personalizadas para evitar o fornecimento de dicas a potenciais atacantes;

5.11.4 Assegurar que o tratamento de erros da aplicação não dependa das configurações padrão do servidor, mas sim de uma lógica definida pela aplicação;

5.11.5 Garantir que a memória seja liberada adequadamente em situações de erro para evitar vazamentos de memória e potenciais vulnerabilidades;

5.11.6 Configurar os controles de segurança para que, na ocorrência de erros lógicos, o acesso seja negado por padrão;

5.11.7 Registrar detalhes de erros internamente para permitir uma análise posterior, mantendo estes registros seguros e acessíveis apenas para pessoal autorizado;

5.11.8 Implementar a análise e o monitoramento de erros para detectar padrões anômalos ou indicativos de tentativas de ataque;

5.11.9 Validar todas as entradas de usuário que possam ser refletidas em mensagens de erro para evitar ataques de injeção;

5.11.10 Estabelecer um sistema de notificação para alertar a equipe de desenvolvimento ou operações sobre erros críticos que possam afetar a segurança;

5.11.11 Implementar testes para garantir que a aplicação possa lidar de forma segura com condições de erro inesperadas;

5.11.12 Ter um plano de resposta a incidentes que inclua procedimentos para lidar com erros graves que possam comprometer a segurança.

Seção X
Do Controle de Log

5.12 O controle de log deverá atender os seguintes requisitos:

5.12.1 Implementar os controles de log em um sistema de confiança, como o servidor, para centralizar e proteger o processo de log;

5.12.2 Assegurar que os sistemas de log capturem tanto eventos de segurança bem-sucedidos quanto os que falharam;

5.12.3 Garantir que os logs registrem eventos significativos para a segurança e operação do sistema;

5.12.4 Certificar-se de que as entradas de log não sejam passíveis de execução como código-fonte, principalmente aquelas contendo dados de fontes não confiáveis;

5.12.5 Limitar o acesso aos logs apenas a pessoal autorizado;

5.12.6 Utilizar uma rotina centralizada para todas as operações de log, a fim de manter a consistência e a segurança;

5.12.7 Evitar armazenar informações sensíveis nos logs, como detalhes do sistema, identificadores de sessão e senhas;

5.12.8 Implementar mecanismos que auxiliem no processo de análise dos logs;

5.12.9 Utilizar funções de hash criptográficas para verificar a integridade dos registros de log;

5.12.10 Implementar medidas para proteger os logs contra alterações não autorizadas, incluindo o uso de armazenamento seguro e imutável para registros críticos;

5.12.11 Definir e cumprir políticas de retenção de logs que equilibrem as necessidades operacionais e de segurança com requisitos legais e regulatórios;

5.12.12 Desenvolver capacidades para respostas automatizadas a eventos específicos identificados nos logs, como alertas de segurança ou indicadores de comprometimento;

5.12.13 Realizar revisões e auditorias periódicas dos logs para identificar padrões anormais ou suspeitos;

5.12.14 Os logs relacionados a autenticação e autorização nos sistemas devem ser armazenados em banco de dados;

5.12.15 Utilizar um timestamp proveniente de um sistema de confiança;

5.12.16 Classificar a gravidade para cada evento;

5.12.17 Destacar eventos de segurança relevantes, caso eles sejam misturados com outros registros de log;

5.12.18 Registar o identificador da conta ou utilizador que causou o evento;

5.12.19 Formalizar e dar transparência aos tipos de log que serão registrados em cada sistema.

5.12.20 Os registros de logs devem incluir:

5.12.20.1 A identificação inequívoca do usuário que acessou os recursos;

5.12.20.2 A natureza do evento, como por exemplo, sucesso ou falha de autenticação, tentativa de troca de senha etc.;

5.12.20.3 A data, hora e fuso horário, observando-se a HLB;

5.12.20.4 O endereço IP (Internet Protocol), porta de origem da conexão, identificador do ativo, coordenadas geográficas, se disponíveis, e outras informações que possam identificar a possível origem do evento.

5.13 Deverá ser registrado em log:

5.13.1 Todas as falhas de validação de entrada e saída de dados;

5.13.2 Todas as falhas de controle de acesso;

5.13.3 Todos os eventos suspeitos de adulteração, inclusive alterações inesperadas no estado dos dados;

5.13.4 As tentativas de ligação com tokens de sessão inválidos ou expirados;

5.13.5 Todas as exceções lançadas pelo sistema;

5.13.6 Todas as funções administrativas, inclusive as mudanças realizadas nas configurações de segurança;

5.13.7 Todas as falhas de ligação TLS com o backend;

5.13.8 Todas as falhas que ocorreram nos módulos de criptografia;

5.13.9 Todos os eventos de autenticação, tanto as bem-sucedidas quanto as malsucedidas;

5.13.10 Todos os acessos a recursos e dados privilegiados;

5.13.11 Todos os acessos e alterações nos registros de auditoria;

5.13.12 A utilização de usuários, perfis e grupos privilegiados;

5.13.13 A inicialização, suspensão e reinicialização de serviços;

5.13.14 As modificações da lista de membros de grupos privilegiados;

5.13.15 As modificações de política de senhas, como, por exemplo, tamanho, expiração, bloqueio automático após exceder determinado número de tentativas de autenticação, histórico etc.;

5.13.16 O acesso ou modificação de arquivos ou de sistemas considerados críticos;

5.13.17 Os eventos obtidos por meio de quaisquer mecanismos de segurança existentes.

Seção XI
Proteção de Dados

5.14 A proteção de dados deverá atender os seguintes requisitos:

5.14.1 Implementar uma política de privilégio mínimo, garantindo que os usuários tenham acesso apenas às funcionalidades, dados e informações do sistema estritamente necessários para realizar suas tarefas;

5.14.2 Assegurar a proteção contra acessos não autorizados de todas as cópias temporárias ou registradas em cache que contenham dados sensíveis no servidor, removendo esses arquivos assim que não forem mais necessários;

5.14.3 Cifrar informações altamente sensíveis armazenadas no servidor, como dados de verificação de autenticação, usando algoritmos criptográficos reconhecidos e seguros;

5.14.4 Salvaguardar o código-fonte no servidor para evitar que seja indevidamente acessado ou baixado por usuários;

5.14.5 Não armazenar senhas, *strings* de conexão ou outras informações confidenciais em texto claro ou de forma criptograficamente insegura no lado cliente;

5.14.6 Eliminar comentários do código de produção que possam revelar detalhes internos do sistema ou outras informações sensíveis;

5.14.7 Remover do sistema aplicações desnecessárias e documentação que possam fornecer informações úteis para atacantes;

5.14.8 Evitar a inclusão de informações sensíveis em parâmetros de pedidos HTTPS GET;

5.14.9 Desabilitar a função de auto completar em formulários que contenham informações sensíveis, incluindo o formulário de autenticação;

5.14.10 Desativar o armazenamento em cache no lado do cliente para páginas que contenham informações sensíveis, utilizando cabeçalhos HTTPS adequados para controlar o cache;

5.14.14 Suportar a remoção efetiva de dados sensíveis do sistema quando eles não forem mais necessários;

5.14.12 Implementar mecanismos de controle de acesso apropriados para dados sensíveis armazenados no servidor, incluindo dados em cache, arquivos temporários e dados restritos a usuários específicos;

5.14.13 Permitir monitoramento e registro de acessos a dados sensíveis para detectar e responder a atividades suspeitas ou não autorizadas;

5.14.14 Proteger APIs e interfaces de dados que permitem o acesso a dados sensíveis, garantindo autenticação, autorização e validação de solicitações;

5.14.15 Quando possível, usar técnicas de anonimização ou pseudonimização para reduzir o risco associado ao armazenamento e processamento de dados sensíveis;

5.14.16 Em ambientes de desenvolvimento e teste, utilizar dados de teste que não contenham informações reais de usuários para evitar exposição acidental de dados sensíveis;

5.14.17 Manter uma segregação clara entre dados de produção e dados de teste, e entre diferentes ambientes operacionais, para prevenir vazamentos de dados e contaminação cruzada;

5.14.18 Realizar auditorias de segurança regulares nos dados armazenados e processados pela aplicação para identificar possíveis vulnerabilidades ou falhas na proteção de dados;

5.14.19 Ao exibir informações sensíveis, usar máscaras para ocultar partes desses dados quando for de acesso público e quando necessário, além das rotinas já utilizadas para dados sigilosos ou restritos.

5.14.20 Implementar medidas para detectar e prevenir vazamentos de dados, como a utilização de soluções de prevenção de perda de dados (DLP);

5.14.21 Ter um plano de resposta a incidentes de segurança de dados para lidar com vazamentos ou violações, incluindo notificação de usuários afetados e autoridades, conforme necessário;

5.14.22 Fornecer formação regular sobre segurança de dados para a equipe de desenvolvimento, para assegurar que estejam cientes das melhores práticas e das últimas ameaças;

5.14.23 Ao exibir informações sensíveis, usar técnicas de ofuscação para proteger os dados, especialmente quando estes precisam ser usados em ambientes menos seguros;

5.14.24 Garantir que existam procedimentos adequados de backup e recuperação de dados para proteger contra a perda de dados devido a falhas do sistema, desastres naturais ou ataques cibernéticos;

5.14.25 Garantir que o consentimento para a coleta e tratamento de dados pessoais seja obtido de maneira clara e explícita, e que os usuários sejam informados sobre como seus dados serão usados;

5.14.26 Facilitar a remoção de dados pessoais a pedido do usuário, em conformidade com a legislação de proteção de dados;

5.14.27 Conduzir Avaliações de Impacto à Proteção de Dados para processos e sistemas que tratam dados pessoais, especialmente para novas tecnologias e práticas;

5.14.28 Estabelecer um plano de resposta a incidentes que inclua notificação às autoridades e aos titulares dos dados no caso de violações de segurança que possam resultar em riscos aos direitos e liberdades dos indivíduos.

Seção XII

Seguranças das Comunicações

5.15 A segurança das comunicações deverá atender os seguintes requisitos:

5.15.1 Utilizar criptografia, como TLS, para a transmissão de todas as informações sensíveis Complementar com a criptografia de arquivos contendo dados sensíveis, especialmente em ligações que não utilizam o protocolo HTTPS;

5.15.2 Garantir que os certificados TLS sejam válidos, contenham o nome de domínio correto, estejam atualizados e sejam acompanhados de certificados intermediários, se necessário;

5.15.3 Configurar o sistema para não fornecer uma ligação insegura em caso de falha nas ligações TLS;

5.15.4 Implementar ligações TLS para todo o conteúdo que exija acesso autenticado ou contenha informações sensíveis;

5.15.5 Utilizar TLS para ligações com sistemas externos que envolvam funções ou informações sensíveis;

5.15.6 Adotar um padrão único de implementação TLS, configurado de maneira apropriada para maximizar a segurança;

5.15.7 Definir claramente a codificação de caracteres para todas as conexões, a fim de evitar problemas de interpretação de dados;

5.15.8 Filtrar informações sensíveis nos parâmetros do "HTTPS referer" ao direcionar para sites externos;

5.15.9 Realizar verificações regulares e auditorias de segurança nas configurações de TLS para garantir que não existam vulnerabilidades conhecidas;

5.15.10 Implementar medidas para proteger contra ataques de Man-in-the-Middle (MitM), como a utilização de HSTS (HTTP Strict Transport Security);

5.15.11 Manter os protocolos de comunicação, como TLS, atualizados para as versões mais recentes e seguras;

5.15.12 Realizar auditorias regulares nos certificados utilizados e nas respectivas cadeias de confiança para garantir a validade e a confiabilidade;

5.15.13 Assegurar que os cabeçalhos HTTPS e outros metadados de comunicação não revelem informações sensíveis ou detalhes do sistema;

5.15.14 Isolar a transmissão de dados sensíveis de outros tipos de dados para reduzir o risco de exposição acidental;

5.15.15 Assegurar que as implementações de criptografia estejam sempre atualizadas para utilizar as versões mais seguras e evitar algoritmos conhecidos por suas vulnerabilidades;

5.15.16 Implementar o DNS Security Extensions (DNSSEC) para proteger contra ataques de redirecionamento e envenenamento de cache DNS, garantindo a autenticidade das respostas aos pedidos de DNS;

5.15.17 Utilizar redes privadas virtuais (VPNs) para garantir conexões seguras para acessos remotos, especialmente em situações onde os servidores e magistrados acessam o sistema de fora da rede corporativa;

5.15.18 Usar WAFs para monitorar e filtrar o tráfego HTTPS entre a internet e a aplicação web, ajudando a proteger contra ataques comuns da web;

5.15.19 Realizar testes de penetração e auditorias de segurança regularmente para identificar e remediar vulnerabilidades nas comunicações e infraestrutura;

5.15.20 Implementar uma gestão rigorosa das configurações de rede para garantir que as medidas de segurança, como firewalls e segmentação de rede, sejam mantidas adequadamente;

5.15.21 Fornecer treinamento regular em segurança de redes para desenvolvedores e administradores de sistemas para garantir que estejam cientes das melhores práticas e das últimas ameaças.

Seção XIII

Configuração do Sistema

5.16 A configuração do sistema deverá atender os seguintes requisitos:

5.16.1 Assegurar que servidores, frameworks e componentes do sistema estejam executando a versão mais recente e segura disponível;

5.16.2 Garantir a aplicação de todas as atualizações de segurança relevantes para a versão em uso de servidores, frameworks e componentes;

5.16.3 Proibir a listagem automática de diretórios nos servidores para evitar a exposição de estruturas de arquivos;

5.16.4 Limitar os privilégios do servidor web, processos e contas de serviços ao mínimo necessário para suas funções;

5.16.5 Garantir que as exceções no sistema sejam tratadas de modo seguro, evitando a exposição de informações sensíveis;

5.16.6 Excluir funcionalidades, arquivos e códigos desnecessários para o funcionamento da aplicação, especialmente no ambiente de produção;

5.16.7 Antes de migrar para o ambiente de produção, remover todo o código de teste ou qualquer funcionalidade que não seja necessária para o funcionamento da aplicação;

5.16.8 Configurar o arquivo "robots.txt" para prevenir a indexação de arquivos sensíveis por robôs de busca, isolando diretórios confidenciais;

5.16.9 Definir e gerenciar quais métodos HTTPS (GET, POST, etc.) a aplicação suportará e como serão tratados nas diferentes páginas;

5.16.10 Desabilitar extensões HTTP que não sejam necessárias, e assegurar mecanismos de autenticação seguros para as extensões utilizadas;

5.16.11 Remover informações desnecessárias dos cabeçalhos de resposta HTTPS que possam revelar detalhes sobre o sistema operacional ou versões de software;

5.16.12 Manter a configuração de segurança da aplicação de forma legível para facilitar a auditoria;

5.16.13 Implementar um sistema de gestão de ativos para manter registros atualizados dos componentes, integrações e softwares;

5.16.14 Separar claramente o ambiente de desenvolvimento da rede de produção e restringir o acesso a grupos específicos;

5.16.15 Implementar um sistema de controle de mudanças para gerenciar e registrar alterações no código, tanto no desenvolvimento quanto na produção;

5.16.16 Empregar ferramentas especializadas para verificar e aplicar configurações de segurança recomendadas em sistemas e aplicações;

5.16.17 Monitorar continuamente as configurações dos sistemas para detectar e responder rapidamente a mudanças não autorizadas;

5.16.18 Manter documentação detalhada das configurações de segurança, incluindo justificativas para quaisquer desvios das configurações padrão ou recomendadas;

5.16.19 Fornecer treinamento específico em segurança para administradores de sistemas para garantir que estejam cientes das melhores práticas de configuração segura.

Seção XIV

Segurança em base de dados

5.17 A segurança em base de dados deverá atender os seguintes requisitos:

5.17.1 Empregar consultas parametrizadas e fortemente tipificadas para prevenir ataques de injeção de SQL;

5.17.2 Implementar validação rigorosa de entrada e codificação (escaping) de saída para lidar com meta caracteres Em caso de falha, os comandos não devem ser executados na base de dados;

5.17.3 Garantir que as variáveis sejam fortemente tipificadas para aumentar a segurança e a integridade dos dados;

5.17.4 Realizar a codificação apropriada de meta caracteres em instruções SQL para mitigar riscos de injeção de SQL;

5.17.5 Assegurar que a aplicação utilize o menor nível de privilégios necessário ao acessar a base de dados;

5.17.6 Utilizar credenciais robustas e seguras para o acesso à base de dados;

5.17.7 Armazenar *strings* de conexão em um arquivo de configuração separado, em um sistema seguro, e cifrar as informações sensíveis;

5.17.8 Utilizar procedimentos armazenados para abstrair o acesso aos dados, permitindo a remoção de permissões diretas sobre as tabelas do banco de dado;

5.17.9 Encerrar a conexão com a base de dados assim que ela não for mais necessária;

5.17.10 Alterar senhas padrão de contas administrativas para senhas robustas ou implementar autenticação de múltiplos fatores. Desativar funcionalidades desnecessárias na base de dados e instalar apenas os componentes necessários;

5.17.11 Eliminar esquemas e bases de dados de exemplo incluídos por padrão pelo fornecedor;

5.17.12 Desativar todas as contas criadas por padrão que não sejam necessárias para os requisitos de negócio;

5.17.13 Conectar à base de dados com diferentes credenciais de segurança para cada tipo de necessidade (usuário, somente leitura, convidado, administrador);

5.17.14 Implementar monitoramento e auditoria contínuos das atividades na base de dados para detectar acessos anômalos ou não autorizados;

5.17.15 Estabelecer procedimentos de backup e recuperação de dados robustos para a base de dados, para proteger contra perda de dados;

5.17.16 Segregar dados sensíveis em diferentes esquemas ou bases de dados, quando possível, para reduzir o risco de acesso não autorizado;

5.17.17 Implementar um controle de acesso baseado em papéis para gerenciar o acesso à base de dados de acordo com as responsabilidades do usuário;

5.17.18 Fornecer treinamento específico em segurança para DBAs para garantir que estejam cientes das melhores práticas em Banco de Dados.

Seção XV

Gestão de Arquivos

5.18 A gestão de arquivos deverá atender os seguintes requisitos:

5.18.1 Evitar passar dados fornecidos pelos usuários diretamente para funções de inclusão dinâmica de arquivos;

5.18.2 Exigir autenticação antes de permitir o carregamento de arquivos;

5.18.3 Limitar os tipos de arquivos que podem ser enviados, aceitando apenas os estritamente necessários para o propósito do negócio;

5.18.4 Assegurar que os arquivos enviados correspondem ao tipo esperado, validando os cabeçalhos do arquivo, além da extensão;

5.18.5 Evitar o armazenamento de arquivos no mesmo diretório da aplicação web. Preferir armazená-los em um servidor de conteúdo dedicado ou na base de dados;

5.18.6 Prevenir ou restringir o carregamento de arquivos que possam ser interpretados ou executados pelo servidor web;

5.18.7 Desativar privilégios de execução em diretórios onde os arquivos são armazenados;

5.18.8 Em ambientes UNIX, implantar o carregamento seguro de arquivos usando montagem de diretório como unidade lógica ou ambiente de "chroot";

5.18.9 Utilizar uma lista branca de nomes e tipos de arquivos permitidos ao referenciar arquivos. Validar o valor do parâmetro passado e rejeitar ou substituir por um valor padrão se não corresponder ao esperado;

5.18.10 Não transmitir dados informados pelo usuário para redirecionamentos dinâmicos sem tratamento. Aceitar apenas URLs relativas e validadas;

5.18.11 Não passar caminhos de diretórios ou arquivos em pedidos. Utilizar um mecanismo de mapeamento para índices definidos em uma lista pré-estabelecida;

5.18.12 Nunca enviar o caminho absoluto do arquivo para o cliente;

5.18.13 Garantir que os arquivos da aplicação e recursos sejam configurados apenas com o atributo de leitura;

5.18.14 Verificar os arquivos submetidos pelos usuários para detectar vírus e malwares;

5.18.15 Implementar procedimentos de backup e recuperação para arquivos críticos, garantindo que possam ser recuperados em caso de perda ou corrupção;

5.18.16 Manter registros de todas as atividades de acesso e modificação de arquivos importantes para auditoria e rastreamento de acesso;

5.18.17 Implementar criptografia robusta para proteger arquivos sensíveis armazenados (em repouso) no sistema;

5.18.18 Implementar controles de acesso baseados em função para restringir quem pode ler, modificar ou excluir arquivos específicos;

5.18.19 Segregar arquivos com diferentes níveis de sensibilidade em diferentes áreas ou sistemas de armazenamento para minimizar o risco de acesso não autorizado;

5.18.20 Implementar monitoramento em tempo real das atividades de arquivo para detecção precoce de comportamento suspeito ou malicioso, como acesso, modificação ou exclusão anormal de arquivos;

5.18.21 Para arquivos críticos, considerar o uso de assinaturas digitais para verificar a integridade e autenticidade dos arquivos, especialmente aqueles que são distribuídos ou compartilhados;

5.18.22 Implementar soluções de segurança que verifiquem os arquivos carregados em busca de malwares, scripts maliciosos ou outros conteúdos prejudiciais;

5.18.23 Utilizar sistemas de controle de versão para gerenciar alterações nos arquivos, permitindo rastrear modificações e restaurar versões anteriores, se necessário;

5.18.24 Definir e implementar uma política de retenção de arquivos para determinar por quanto tempo os arquivos devem ser mantidos e quando devem ser seguramente destruídos ou arquivados;

5.18.25 Realizar auditorias regulares para garantir que os arquivos sensíveis estão sendo armazenados, acessados e gerenciados conforme as políticas de segurança;

5.18.26 Proporcionar formação específica sobre segurança de arquivos para os membros da equipes, ensinando as melhores práticas para o manuseio de dados sensíveis;

5.18.27 Implementar autenticação forte, como autenticação de dois fatores, para acesso a arquivos sensíveis, especialmente em ambientes de armazenamento em nuvem ou compartilhados;

5.18.28 Para dados altamente sensíveis, considerar o armazenamento em containers seguros ou sistemas de gestão de documentos com funcionalidades de segurança avançadas;

5.18.29 Implementar verificações de tamanho e formato de arquivo durante o carregamento para prevenir ataques como o carregamento de arquivos grandes que podem causar negação de serviço ou arquivos em formatos inesperados que podem conter código malicioso.

Seção XVI

Gestão de Memória

5.19 A gestão de memória deverá atender os seguintes requisitos:

5.19.1 Implementar um controle rigoroso para a entrada e saída de dados não confiáveis, evitando corrupções de memória e vulnerabilidades;

5.19.2 Assegurar que os buffers sejam suficientemente grandes para os dados que eles devem armazenar, evitando problemas de buffer overflow;

5.19.3 Quando realizar chamadas de função em ciclos, verificar cuidadosamente os limites do buffer para prevenir escritas de dados além do espaço alocado;

5.19.4 Truncar todas as strings de entrada para um tamanho razoável antes de passá-las para funções de cópia e concatenação para evitar overflow;

5.19.5 Ao liberar recursos reservados (como objetos de ligação, identificadores de arquivos), não depender apenas do garbage collector e realizar a liberação de forma explícita;

5.19.6 Utilizar pilhas não executáveis, quando disponíveis, para prevenir execuções de código malicioso através de overflow;

5.19.7 Evitar o uso de funções conhecidas por suas vulnerabilidades, como printf(), strcat(), strcpy(), e preferir alternativas mais seguras;

5.19.8 Garantir a liberação adequada da memória alocada ao final de sub-rotinas (funções/métodos) e em todos os pontos de saída;

5.19.9 Utilizar ferramentas de análise de memória, como Valgrind ou AddressSanitizer, durante o desenvolvimento para identificar vazamentos de memória e outros problemas relacionados;

5.19.10 Realizar revisões de código e testes focados em potenciais problemas de gestão de memória, como vazamentos, acesso a memória não inicializada e uso após liberação;

5.19.11 Inicializar variáveis apropriadamente ao declará-las para evitar o uso de valores indeterminados que podem levar a comportamentos imprevisíveis;

5.19.12 Providenciar treinamento específico para desenvolvedores sobre gestão segura de memória, destacando práticas comuns e erros a evitar;

5.19.13 Implementar um tratamento robusto de exceções e erros relacionados à memória, garantindo que o software se comporte de maneira previsível e segura em situações de falha de alocação de memória;

5.19.14 Separar e proteger dados sensíveis (como chaves criptográficas e informações pessoais) na memória, utilizando técnicas como alocação de memória dedicada e limpeza de dados sensíveis após o uso;

5.19.15 Monitorar o consumo de memória da aplicação para identificar possíveis vazamentos ou uso excessivo de memória que possam indicar problemas de segurança ou estabilidade;

5.19.16 Preferir o uso de bibliotecas e frameworks que ofereçam abstrações seguras e eficientes para a gestão de memória, minimizando a possibilidade de erros;

5.19.17 Manter todas as dependências e bibliotecas de terceiros atualizadas, pois as versões mais recentes geralmente incluem correções para vulnerabilidades conhecidas relacionadas à gestão de memória;

5.19.18 Em linguagens de programação de baixo nível, como C e C++, seguir práticas seguras como evitar o uso direto de funções potencialmente perigosas (malloc, free, etc.) e preferir abstrações modernas ou bibliotecas que ofereçam maior segurança;

5.19.19 Aplicar padrões de design que promovam a gestão segura de memória;

5.19.20 Realizar auditorias de segurança específicas focadas em identificar problemas de gestão de memória, especialmente em partes críticas do sistema.

Seção XVII

Segurança no repositório de código-fonte

5.20 A segurança no repositório de código-fonte deverá atender os seguintes requisitos:

5.20.1 Implementar um controle de acesso rigoroso ao repositório de código-fonte. Apenas desenvolvedores autorizados e relevantes para cada projeto devem ter acesso Utilizar autenticação forte, preferencialmente com autenticação de dois fatores (2FA);

5.20.2 Estabelecer um processo de revisão de código onde merge requests (solicitações de fusão) são minuciosamente revisadas por outros desenvolvedores antes da integração no repositório principal. Isso ajuda a identificar potenciais vulnerabilidades ou más práticas de programação;

5.20.3 Manter um registro de todas as mudanças feitas no repositório, incluindo quem fez a mudança, o que foi mudado, e quando a mudança ocorreu. Utilizar ferramentas de auditoria para monitorar e revisar o acesso e alterações ao repositório;

5.20.4 Utilizar branches protegidos para evitar alterações diretas no código-fonte principal ou em branches críticos, como a branch de produção;

5.20.5 Assegurar backups regulares do repositório de código-fonte e ter um plano de recuperação em caso de perda de dados ou corrupção;

5.20.6 Empregar assinaturas digitais para verificar a autenticidade das alterações submetidas ao repositório;

5.20.7 Proteger o armazenamento físico e virtual do repositório de código-fonte contra acessos não autorizados e ataques cibernéticos;

5.20.8 Integrar ferramentas de análise estática e dinâmica de segurança do código para identificar automaticamente vulnerabilidades antes que o código seja integrado ao repositório principal;

5.20.9 Assegurar que os processos de Integração Contínua e Entrega Contínua incluam verificações de segurança para prevenir a introdução de vulnerabilidades no código;

5.20.10 Monitorar e gerenciar as dependências de terceiros utilizadas no código-fonte para garantir que estão atualizadas e não apresentam vulnerabilidades conhecidas.

Seção XVIII

Práticas Gerais de Programação

5.21 As práticas gerais de programação deverá atender os seguintes requisitos:

5.21.1 Priorizar a utilização de código bem testado, gerido e aprovado para tarefas comuns, evitando a criação de novas implementações não verificadas;

5.21.2 Utilizar APIs confiáveis para realizar operações do sistema operativo, evitando a execução direta de comandos pelo sistema, especialmente através de shells de comando iniciadas pela aplicação;

5.21.3 Implementar mecanismos de verificação de integridade, como checksum ou hash, para assegurar a integridade de código interpretado, bibliotecas, executáveis e arquivos de configuração;

5.21.4 Empregar mecanismos de bloqueio para prevenir pedidos simultâneos conflitantes e mecanismos de sincronização para evitar condições de concorrência (race conditions);

5.21.5 Assegurar que as variáveis e recursos compartilhados sejam protegidos contra acessos concorrentes inapropriados;

5.21.6 Instanciar explicitamente todas as variáveis e dados persistentes, seja durante a declaração ou antes da primeira utilização;

5.21.7 Quando necessário executar a aplicação com privilégios elevados, aumentar os privilégios o mais tarde possível e revogá-los assim que não forem mais necessários;

5.21.8 Ter cuidado com erros decorrentes de representações internas de números, incluindo tamanho de byte, precisão, distinções de sinal, truncamento, conversões e tratamento de números extremamente grandes ou pequeno;

5.21.9 Não transferir dados fornecidos pelo usuário diretamente para funções de execução dinâmica sem um tratamento adequado dos dados;

5.21.10 Restringir a geração e a alteração de código por parte dos utilizadores;

5.21.11 Rever todas as aplicações secundárias, códigos e bibliotecas de terceiros para determinar a necessidade do negócio e validar as funcionalidades de segurança, uma vez que estas podem introduzir novas vulnerabilidades;

5.21.12 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 cifrados para transferir o código a partir do host do servidor;

5.21.13 Aplicar o princípio do menor privilégio em todo o código, garantindo que processos, usuários e sistemas operem com o mínimo de permissões necessárias para realizar suas funções;

5.21.14 Para operações potencialmente perigosas ou não confiáveis, considerar o uso de técnicas de *sandboxing* ou ambientes virtualizados para isolar essas operações do restante do sistema;

5.21.15 Praticar a codificação defensiva, antecipando cenários de falha e comportamentos inesperados do usuário ou do sistema, e codificando para prevenir esses riscos;

5.21.16 Realizar testes de regressão de segurança regularmente para garantir que as alterações no código não introduzam novas vulnerabilidades;

5.21.17 Estabelecer ambientes de homologação que simulem com precisão o ambiente de produção para testar a segurança antes do lançamento;

5.21.18 Implementar medidas para detectar e prevenir vazamentos de dados sensíveis, tanto no código quanto na execução do software;

5.21.19 Estabelecer um processo de feedback e melhoria contínua para as práticas de segurança, incorporando lições aprendidas, novas ameaças e evoluções tecnológicas.

6 MONITORAMENTO E AUDITORIA

6.1 Por motivos de segurança, os logs (acesso, auditoria, vulnerabilidade, etc.) dos sistemas desenvolvidos e obtidos serão mantidos pela Secretaria de Tecnologia da Informação e Comunicação por, no mínimo, 6 (seis) meses e, no máximo, 12 (doze) meses.

6.2 Em caso de indícios de descumprimento das diretrizes previstas neste normativo, o Grupo Gestor Permanente de Tratamento e Resposta a Incidentes de Segurança Cibernética poderá de ofício ou por determinação do Comitê Gestor de Segurança da Informação e Cibernética realizar auditoria sobre os fatos.

6.3 Os relatórios decorrentes das auditorias ordinárias e extraordinárias realizadas pelo Grupo Gestor Permanente de Tratamento e Resposta a Incidentes de Segurança Cibernética serão encaminhados ao Comitê Gestor de Segurança da Informação, para análise e deliberação.

7 DISPOSIÇÃO FINAL

7.1 O disposto na presente norma será atualizado sempre que alterados os procedimentos e controles ou quando necessário, mediante iniciativa do CGSI.