



**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

## Sumário

1.	OBJETIVO .....	2
2.	ÂMBITO DE APLICAÇÃO .....	2
3.	DEFINIÇÕES .....	2
4.	DOCUMENTOS DE REFERÊNCIA.....	3
5.	RESPONSABILIDADES.....	3
6.	REGRAS BÁSICAS .....	4
7.	CONTROLE DE REGISTROS .....	16
8.	ANEXOS.....	17
9.	REGISTRO DE ALTERAÇÕES.....	19



**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

## 1. OBJETIVO

Este documento tem por objetivo definir um conjunto de boas práticas de codificação segura. As recomendações serão apresentadas em um formato de checklist, que podem ser integradas ao ciclo de desenvolvimento de aplicações.

### 1.1. INTRODUÇÃO

O objetivo da segurança em aplicações é manter a confidencialidade, integridade e disponibilidade dos recursos de informação a fim de permitir que as operações de negócios sejam bem-sucedidas. Este guia se concentra nos controles técnicos específicos para mitigar as ocorrências de vulnerabilidades mais comuns de software.

## 2. ÂMBITO DE APLICAÇÃO

### 2.1. Empresa

Todas as empresas do Grupo CPFL;

### 2.2. Área

EI – DIRETORIA TECNOLOGIA DA INFORMAÇÃO  
SSIO - GERENCIA SERVICOS DE INFRAESTRUTURA E OPERAÇÃO DE TI  
SSMS - GERENCIA SERVICOS E MANUTENÇÃO DE SISTEMAS DE TI

## 3. DEFINIÇÕES

### 3.1 Confidencialidade

Garantia de que os dados da empresa não estarão disponíveis nem serão divulgados a indivíduos, entidades ou processos sem autorização.

### 3.2 Controles

Refere-se a controles tecnológicos ou de processos que podem impedir um criminoso de ler ou usar os dados acessados ou adquiridos. Definem os objetivos principais de uma implementação de segurança apropriada.

### 3.3 Notificação de violação

Refere-se geralmente às leis que se aplicam a uma entidade que exige que tal entidade notifique indivíduos quando ocorrida acesso não autorizado a dados pessoais ou se acredita que tenha ocorrido ou conforme previsto em tais leis.

### 3.4 Impacto

Consequências que podem ocorrer caso o risco se manifeste.

### 3.5 Grupo CPFL

A CPFL Energia S.A., e todas as suas subsidiárias e/ou afiliadas, exceto as empresas com seus próprios padrões de governança e gestão que compartilham controle com outras empresas.

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	2 de 20



**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

#### 4. DOCUMENTOS DE REFERÊNCIA

##### 4.1. Internos

- GED 14369 – Diretrizes de Segurança da Informação.
- GED 19271 - Requisitos de Segurança para Projetos.
- GED 18883 - Norma de Desenvolvimento Seguro

##### 4.2. Externos

Sans and TippingPoint "The Top Cyber Security Risks"

<http://www.sans.org/top-cyber-security-risks/>

- Web Application Security Consortium  
<http://www.webappsec.org/>
- Common Weakness Enumeration (CWE)  
<http://cwe.mitre.org/>
- Department of Homeland Security  
Build Security In Portal  
<https://buildsecurityin.us-cert.gov/daisy/bsi/home.html>
- CERT Secure Coding  
<http://www.cert.org/secure-coding/>
- MSDN Security Developer Center  
<http://msdn.microsoft.com/en-us/security/default.aspx>
- SQL Injection Cheat Sheet  
<http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- Cross Site Scripting (XSS) Cheat Sheet  
<http://hackers.org/xss.html>

#### 5. RESPONSABILIDADES

##### 5.1 DIRETORIA TECNOLOGIA DA INFORMAÇÃO

- Monitorar e reforçar a importância de cumprimento desse checklist junto aos seus colaboradores que estão sob sua responsabilidade;
- Cumprir as diretrizes definidas neste checklist

##### 5.2 GERENCIA SERVICOS DE INFRAESTRUTURA E OPERAÇÃO DE TI

- Monitorar e reforçar a importância de cumprimento desse checklist junto aos seus colaboradores que estão sob sua responsabilidade;
- Cumprir as diretrizes definidas neste checklist

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	3 de 20



**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

### 5.3 GERENCIA SERVICOS E MANUTENÇÃO DE SISTEMAS DE TI

- Monitorar e reforçar a importância de cumprimento desse checklist junto aos seus colaboradores que estão sob sua responsabilidade;
- Cumprir as diretrizes definidas neste checklist

## 6. REGRAS BÁSICAS

Construir software seguro exige ter o conhecimento básico dos princípios de segurança. Uma revisão abrangente dos princípios de segurança está fora do escopo deste guia, porém os conceitos de segurança serão abordados de forma superficial e abrangente. O objetivo da segurança em aplicações é manter a confidencialidade, integridade e disponibilidade dos recursos de informação a fim de permitir que as operações de negócios sejam bem-sucedidas.

Esse objetivo é alcançado através da implementação de controles de segurança. Este guia se concentra nos controles técnicos específicos para mitigar as ocorrências de vulnerabilidades mais comuns de software. Como o foco principal são as aplicações web e sua infraestrutura de apoio, boa parte deste guia pode ser usado para qualquer plataforma de desenvolvimento de software.

É importante que as equipes de desenvolvimento de aplicações entendam que os controles do lado cliente, como validação de entrada de dados no cliente, campos ocultos e controles de interface (combo box, radio buttons), fornecem pouco ou nenhum benefício de segurança. Neste caso, um atacante pode usar ferramentas como proxies do lado do cliente, como o OWASP WebScarab, Burp ou ferramentas de captura de pacotes de rede, como o Wireshark, para analisar o tráfego de aplicativo e enviar requisições manipuladas, burlando todas as interfaces. Além disso, o Flash, os Applets Java e demais objetos que trabalham no lado cliente pode ser compilados e analisados em busca de falhas.

As falhas de segurança de software podem ser introduzidas em qualquer fase do ciclo de desenvolvimento de software, inclusive:

- Não identificar as necessidades de segurança no início.
- Criação de arquiteturas conceituais que possuem erros de lógica.
- Usando práticas ruins de codificação que introduzem vulnerabilidades técnicas.
- Implantação do software de modo inapropriado.
- Inserção de falhas durante a manutenção ou atualização.

Além disso, é importante entender que as vulnerabilidades de softwares podem ter um escopo muito além do próprio software. Dependendo da natureza do software, da vulnerabilidade e da infraestrutura de apoio, os impactos de uma exploração bem-sucedida podem comprometer qualquer ou todos os seguintes aspectos:

- O software e sua informação associada.
- O sistema operacional dos servidores associados.

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	4 de 20



**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

- Banco de dados de backend.
- Outras aplicações em um ambiente compartilhado.
- O sistema do usuário.
- Outros softwares que o usuário interage

## 6.1. Checklist de Práticas de Codificação Segura

### 6.1.2. Validação de Entrada de Dados:

1. Efetuar toda a validação dos dados em um sistema confiável, por exemplo: centralizar todo controle no servidor.
2. Identificar todas as fontes de dados e classificar as fontes como confiável/não confiável. Em seguida, validar os dados provenientes de fontes não confiáveis (ex: banco de dados, stream de arquivos etc.).
3. A rotina de validação de dados de entrada deve ser centralizada na aplicação.
4. Especificar conjunto de caracteres apropriados, como UTF-8, para todas as fontes de entrada de dados.
5. Codificar os dados para um conjunto de caracteres comuns antes da validação (Canonicalize).
6. Quando há falha de validação a aplicação deve rejeitar os dados fornecidos.
7. Determinar se o sistema suporta conjuntos de caracteres estendidos UTF-8 e em caso afirmativo, validar após efetuar a decodificação UTF-8.
8. Validar todos os dados provenientes dos clientes antes do processamento, incluindo todos os parâmetros, campos de formulário, conteúdo das URLs e cabeçalhos HTTP, por exemplo: nomes e valores dos Cookies. Certificar-se também de incluir automaticamente mecanismos de postback2 nos trechos de código JavaScript, Flash ou qualquer outro código incorporado.
9. Verificar os valores de cabeçalho, tanto das requisições, como das respostas, que contém apenas caracteres ACII.
10. Validar dados provenientes de redirecionamentos. Os atacantes podem incluir conteúdo malicioso diretamente para o alvo do mecanismo de redirecionamento, podendo assim contornar a lógica da aplicação e qualquer validação executada antes do redirecionamento.
11. Validar tipos de dados esperados.
12. Validar intervalo de dados.
13. Validar o comprimento dos dados.
14. Validar, sempre que possível, todos os dados de entradas através de um método baseado em "lista branca" que utiliza uma lista de caracteres ou expressão regular que define os caracteres permitidos.
15. Se qualquer caractere potencialmente 'perigoso' precisa ser permitido na entrada de dados da aplicação, certifique-se que foram implementados controles adicionais como codificação 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: <, >, ", ', %, (, ), &, +, \, \', \".

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	5 de 20



**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

16. Se a rotina de validação padrão não aborda as seguintes entradas, então elas devem ser verificadas discretamente:
17. Verificar bytes nulos (00%).
18. Verificar se há caracteres de nova linha (%0d, %0a, \r, \n).
19. Verificar se há caracteres ponto-ponto barra (../ ou ..\ ) que alteram caminhos. Nos casos de conjunto de caracteres que usam 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 encoding3 ) ou outras formas de ataques por ofuscação.

### Codificação da Saída de Dados:

1. Efetuar toda a codificação dos dados em um sistema confiável, por exemplo: centralizar todo controle no servidor.
2. Utilizar uma rotina padrão, testada, para cada tipo de codificação de saída
3. Realizar a codificação, baseada em contexto, de todos os dados retornados para o cliente que originam-se de ambiente fora dos limites de confiança da aplicação. A codificação da entidade HTML é um exemplo, mas nem sempre funciona para todos os casos.
4. Codificar todos os caracteres a menos que sejam conhecidos por serem seguros para o interpretador de destino.
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.
6. Tratar todos os dados provenientes de fontes não confiáveis que geram comandos de sistema operacional.

### Autenticação e Gerenciamento de Senhas:

1. Requerer autenticação para todas as páginas e recursos, exceto para aqueles que são intencionalmente públicos.
2. Os controles de autenticação devem ser executados em um sistema confiável, por exemplo: centralizar todo controle no servidor.
3. Estabelecer e utilizar serviços de autenticação padronizados e testados, sempre que possível.
4. Utilizar uma implementação centralizada para realizar os controles de autenticação, disponibilizando bibliotecas que invocam os serviços de autenticação externos.
5. Separar a lógica de autenticação do recurso que está sendo requisitado e usar redirecionadores dos controles de autenticação centralizados.
6. Quando situações excepcionais ocorrerem nos controles de autenticação, executar procedimentos em caso de falha de modo a manter o sistema seguro.
7. Todas as funções administrativas e de gerenciamento de contas devem ser tão seguras quanto o mecanismo de autenticação principal.
8. Se a aplicação gerencia um repositório de credenciais, esta deverá garantir que as senhas sejam armazenadas no banco de dados somente o resumo/hash da senha na forma de one-way salted

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	6 de 20



**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

hashes4, e que a tabela/arquivo que armazena as senhas e que as chaves sejam manipuladas apenas pela aplicação. Obs.: não utilizar o algoritmo de hash MD5, caso seu uso puder ser evitado.

9. A geração dos resumos (hash) das senhas devem ser executadas em um sistema confiável, por exemplo: centralizar o controle no servidor.
10. Validar os dados de autenticação somente ao término de todas as entradas de dados, especialmente para as implementações de autenticação sequencial.
11. As respostas de falhas de autenticação não devem indicar qual parte dos dados de autenticação estão incorretos. Por exemplo: em vez de exibir mensagens como “Nome de usuário incorreto” ou “Senha incorreta”, apenas utilize mensagens como: “Usuário e/ou senha inválidos”, para ambos os casos de erro. As respostas de erro devem ser literalmente idênticas nos dois casos.
12. Utilize autenticação para conexão a sistemas externos que envolvem tráfego de informação sensível ou acesso a funções.
13. As credenciais de autenticação para acessar serviços externos à aplicação devem ser criptografados e armazenados em um local protegido em um sistema confiável, por exemplo: no servidor da aplicação. Obs.: o código fonte não é considerado um local seguro.
14. Utilizar apenas requisições POST para transmitir credenciais de autenticação.
15. Somente trafegar senhas (não temporárias) através de uma conexão criptografada (SSL/TLS) ou como dado criptografado, como no caso de envio de e-mail criptografado. Senhas temporárias enviadas por e-mail podem ser um caso de exceção aceitável.
16. Exigir que os requisitos de complexidade de senha estabelecidos pela política ou regulamento sejam cumpridos. As credenciais de autenticação devem ser suficientes para resistir ataques que tipicamente ameaçam o ambiente de produção. Um exemplo pode ser a exigência do uso simultâneo de caracteres alfabéticos, numérico e/ou caracteres especiais.
17. Exigir que os requisitos de comprimento de senha estabelecidos pela política ou regulamento sejam cumpridos. O uso de oito caracteres é o mais comum, porém 16 é melhor ou então considere o uso de senhas que contém várias palavras (uma frase).
18. A entrada de senha deve ser ocultada na tela do usuário. Em HTML, utilize o campo tipo “password”.
19. Desativar a conta após um número pré-definido de tentativas inválidas de login (ex: cinco tentativas é o mais comum). A conta deve ser desativada por um período de tempo suficiente para desencorajar a dedução das credenciais pelo método de força bruta, mas nem tão longo ao ponto de permitir um ataque de negação de serviço.
20. 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.
21. Esquemas de pergunta/resposta (pré-definidas) usadas para a redefinição de senha devem evitar ataques que lançam respostas aleatórias, ex “livro favorito” é uma questão ruim, pois “A Bíblia” é uma resposta muito comum.
22. Se for usar redefinição de senha baseada em e-mail, somente envie um e-mail para um endereço pré-definido contendo um link ou senha de acesso temporário que permitem ao usuário redefinir a senha.
23. O tempo de validade das senhas e dos links temporários devem ser curtos.

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	7 de 20





**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

24. Exigir a mudança de senhas temporárias na próxima vez que o usuário realizar a autenticação no sistema.
25. Notificar o usuário quando a sua senha for reiniciada (reset).
26. Prevenir a reutilização de senhas.
27. As senhas devem ter pelo menos um dia de duração antes de poderem ser alteradas para evitar ataques de reuso de senhas.
28. Garantir que a troca de senhas estejam 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 devem ser controladas administrativamente.
29. Desabilitar a funcionalidade de lembrar a senha nos campos de senha do navegador.
30. A data/hora da última utilização (bem ou malsucedida) de uma conta de usuário deve ser comunicada ao usuário no seu próximo login.
31. Realizar monitoramento para identificar ataques contra várias contas de usuário, utilizando a mesma senha. Este padrão de ataque é utilizado para explorar o uso de senhas padrão.
32. Modificar todas as senhas que por padrão são definidas pelos fornecedores, bem como os identificadores de usuários (IDs) ou desabilite as contas associadas.
33. Exigir uma re-autenticação dos usuários antes da realização de operações críticas.
34. Utilizar autenticação de fatores múltiplos (utilizando simultaneamente token, senha, biometria, etc5) para contas altamente sensíveis ou de alto valor transacional.
35. Caso for utilizar código de terceiros para realizar a autenticação, inspecione cuidadosamente o código para garantir se o mesmo não é afetado por qualquer código malicioso.

### Gerenciamento de Sessão:

1. Utilize controles de gerenciamento de sessão baseados no servidor ou em framework. A aplicação deve reconhecer apenas os identificadores de sessão como válidos.
2. A criação dos identificadores de sessão deve ser sempre realizada em um sistema confiável, por exemplo: centralizar todo controle no servidor.
3. Usar algoritmos bem controlados que garantam a aleatoriedade dos identificadores de sessão.
4. Defina o domínio e o caminho para os cookies que contém identificadores de sessão autenticados para um valor devidamente restrito para o site.
5. A funcionalidade de logout deve encerrar completamente a sessão ou conexão associada.
6. A funcionalidade de logout deve estar disponível em todas as páginas que requerem autenticação.
7. Estabelecer um tempo de expiração da sessão que seja o mais curto possível, baseado no balanceamento dos riscos e requisitos funcionais do negócio. Na maioria dos casos não deve ser mais do que algumas horas.
8. Não permitir logins persistentes (sem prazo para expirar sessão) e realizar o encerramento da sessão periodicamente, mesmo quando a sessão estiver ativa. Especialmente para aplicações que suportam várias conexões de rede ou que se conectam a sistemas críticos. O tempo de

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	8 de 20





**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

encerramento deve apoiar os requisitos de negócio, enquanto o usuário deve receber notificação suficientes para atenuar os impactos negativos destas medidas.

9. Se uma sessão estava estabelecida antes do login, então esta sessão deve ser encerrada para que uma nova sessão seja estabelecida após o login.
10. Gerar um novo identificador de sessão quando houver alguma nova autenticação.
11. Não permitir conexões simultâneas com o mesmo identificador de usuário.
12. Não expor os identificadores de sessão em URLs, mensagens de erro ou logs. Os identificadores de sessão devem apenas serem localizados no cabeçalho do cookie HTTP. Por exemplo, não trafegar os identificadores de sessão na forma de parâmetros GET.
13. Proteger os dados de sessão do lado servidor contra acessos não autorizados, por outros usuários do servidor, através da implementação de controle de acesso apropriado no servidor.
14. Gerar um novo identificador de sessão e desativar o antigo identificador periodicamente. Isto pode mitigar certos cenários de ataques de sequestro de sessão (session hijacking), quando o identificador de sessão original é comprometido.
15. Gerar um novo identificador de sessão caso a segurança da conexão mude de HTTP para HTTPS, como pode ocorrer durante a autenticação. Internamente à aplicação, é recomendável utilizar HTTPS de forma constante em vez de alternar entre HTTP para HTTPS.
16. Utilize mecanismos complementares ao mecanismo de gerenciamento de sessão padrão para operações sensíveis do lado do servidor, como é o caso de operações de gerenciamento de contas, através da utilização de tokens aleatórios ou parâmetros associados à sessão. Este método pode ser usado para prevenir-se de ataques do tipo Cross Site Request Forgery.
17. Utilize mecanismos complementares ao gerenciamento de sessão para operações altamente sensíveis ou críticas utilizando tokens aleatórios ou parâmetros em cada requisição.
18. Utilizar somente identificadores de sessão gerados pelo sistema para gerenciamento de sessão do lado cliente. Evite usar parâmetros ou outros dados fornecidos pelos clientes para o gerenciamento do estado.
19. Configurar o atributo "secure" para cookies transmitidos através de uma conexão TLS.
20. Configurar os cookies com o atributo HttpOnly, a menos que seja explicitamente necessário ler ou definir os valores dos cookies através de scripts do lado cliente da aplicação.

### Controle de Acessos:

1. Utilizar apenas objetos do sistema que sejam confiáveis, como ocorre com os objetos de sessão do servidor, para realizar a tomada de decisões de autorização de acesso.
2. Utilize um único componente em todo o site para realizar o processo de verificação de autorização de acesso. Isso inclui bibliotecas que invocam os serviços externos de autorização.
3. Quando ocorrer alguma falha no controle de acesso elas devem ocorrer de modo seguro.
4. Negar todos os acessos caso a aplicação não consiga ter acesso as informações contidas na configuração de segurança.
5. Garantir o controles de autorização em todas as requisições, inclusive em scripts do lado do servidor, "includes" e requisições provenientes de tecnologias do lado cliente, como AJAX e Flash.

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	9 de 20



**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

6. Separar os trechos de código que contém a lógica privilegiada da aplicação do restante do código da aplicação.
7. Restringir o acesso aos arquivos e outros recursos, incluindo aqueles que estão fora do controle direto da aplicação, somente a usuários autorizados.
8. Restringir o acesso às URLs protegidas somente aos usuários autorizados.
9. Restringir o acesso às funções protegidas somente aos usuários autorizados.
10. Restringir o acesso às referências diretas aos objetos somente aos usuários autorizados.
11. Restringir o acesso aos serviços somente aos usuários autorizados.
12. Restringir o acesso aos dados da aplicação somente aos usuários autorizados.
13. Restringir o acesso aos atributos e dados dos usuários, bem como informações das políticas usadas pelos mecanismos de controle de acesso.
14. Restringir o acesso às configurações de segurança relevantes apenas para usuários autorizados.
15. As regras de controle de acesso representados pela camada de apresentação devem coincidir com as regras presentes no lado servidor.
16. Se o estado dos dados deve ser armazenado no lado cliente, utilize mecanismos de criptografia e verificação de integridade no lado servidor para detectar possíveis adulterações no estado dos dados.
17. Garantir que os fluxos lógicos da aplicação respeitem as regras de negócio.
18. Limitar o número de transações que um único usuário ou dispositivo podem executar em um determinado período. As transações por período devem estar acima da necessidade real do negócio, mas abaixo o suficiente para impedir ataques automatizados.
19. Use o campo "referer" do cabeçalho somente como forma de verificação suplementar. Ele não deve ser usado sozinho como forma de checagem de autorização, pois o valor deste campo pode ser adulterado.
20. Se é permitido a permanência de sessões autenticadas por longos períodos, faça revalidação periódica da autorização do usuário para garantir que seus privilégios não foram modificados e caso forem, realize o registro em log do usuário e exija nova autenticação.
21. Implementar a auditoria das contas de usuário e assegure a desativação de contas não utilizadas. Por exemplo: após não mais do que 30 dias após expirar a senha da conta, ela deve ser desativada.
22. A aplicação deve dar suporte a desativação de contas e encerramento das sessões quando encerrar a autorização do usuário, por exemplo: quando ocorrem alterações de mudança de papéis de usuário, situação profissional, processos de negócio, etc.
23. As contas de serviço ou contas de suporte a conexões provenientes ou destinadas a serviços externos devem possuir o menor privilégio possível.
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 de acesso 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.

#### Práticas de Criptografia:

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	10 de 20


**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

1. Todos as funções de criptografia utilizados para proteger dados sensíveis dos usuários da aplicação devem ser implementados em um sistema confiável (neste caso o servidor).
2. A senha mestre deve ser protegida contra acessos não autorizados.
3. Quando ocorrer alguma falha dos módulos de criptografia, permitir que as falhas ocorram de modo seguro.
4. Todos os números aleatórios, nomes de arquivos aleatórios, GUIDs aleatórios, e strings aleatórias devem ser geradas usando um módulo criptográfico com gerador de números aleatórios aprovado somente se os valores aleatórios gerados forem impossíveis de serem deduzidos.
5. Os módulos de criptografia usados pela aplicação devem ser compatíveis com a FIPS 140-2 ou padrão equivalente (<http://csrc.nist.gov/groups/STM/cmvp/validation.html>)
6. Estabelecer e utilizar uma política e processo que define como é realizado o gerenciamento das chaves de criptografia.

#### Tratamento de Erros e Log:

1. Não exponha informações sensíveis nas repostas de erros, inclusive detalhes de sistema, identificadores de sessão ou informação da conta do usuário.
2. Use mecanismos de tratamento de erros que não exibam informações de debug ou informações da pilha de exceção.
3. Implemente mensagens de erro genéricas e páginas de erro personalizadas.
4. A aplicação deve tratar seus erros sem confiar nas configurações do servidor.
5. A memória alocada deve ser liberada de modo apropriado quando ocorrerem condições de erro.
6. O tratamento de erros lógicos associados com controles de segurança deve por padrão negar o acesso.
7. Todos os controles de log devem ser implementados em um sistema confiável (neste caso o servidor).
8. Os controles de log devem dar suporte tanto para os casos de sucesso ou falha relacionados a eventos de segurança específicos.
9. Garantir que os logs armazenam eventos importantes.
10. Garantir que as entradas de log que incluem dados não confiáveis não sejam executadas como um código na interface de visualização de logs.
11. Restringir o acesso aos logs apenas para pessoal autorizado.
12. Utilizar uma rotina centralizada para realizar todas as operações de log.
13. Não armazenar informações sensíveis nos registros de logs, como detalhes desnecessários do sistema, identificadores de sessão e senhas.
14. Garantir o uso de algum mecanismo que conduza (ou facilite) o processo de análise de logs.
15. Registrar em log todas as falhas de validação de entrada de dados.
16. Registrar em log todas as tentativas de autenticação, especialmente as falhas de autenticação.
17. Registrar em log todas as falhas de controle de acesso.
18. Registrar em log todos os eventos aparentes de adulterações, inclusive alterações inesperadas no estado dos dados.

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	11 de 20



**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

19. Registrar em log as tentativas de conexão com tokens de sessão inválidos ou expirados.
20. Registrar em log todas as exceções lançadas pelo sistema.
21. Registrar em log todas as funções administrativas, inclusive as mudanças realizadas nas configurações de segurança.
22. Registrar em log todas as falhas de conexão TLS com o backend.
23. Registrar em log todas as falhas que ocorreram nos módulos de criptografia.
24. Utilizar uma função de hash criptográfica para validar a integridade dos registros de log.

### Proteção de Dados:

1. Implementar política de privilégio mínimo, restringindo os usuários apenas às funcionalidades, dados e informações do sistema que são necessárias para executar suas tarefas.
2. Proteger todas as cópias temporárias ou registradas em cache que contenham dados sensíveis e estejam armazenados no servidor contra acesso não autorizado e realizar a remoção destes arquivos tão logo não sejam mais necessários.
3. Criptografar informações altamente sensíveis quando armazenadas, como dados de verificação de autenticação, mesmo que estejam no lado servidor. Sempre usar algoritmos bem controlados. Consulte a seção que trata sobre “Práticas de Criptografia” para orientações adicionais.
4. Proteger o código-fonte presente no servidor para que não sejam baixados por algum usuário.
5. Não armazenar senhas, strings de conexão ou outras informações confidenciais em texto claro ou em qualquer forma criptograficamente insegura no lado cliente. Isso vale também quando há incorporação de formatos inseguros como: MS viewstate, Adobe Flash ou código compilado que roda no lado cliente.
6. Remover comentários do código de produção que são acessíveis pelos usuários e podem revelar detalhes internos do sistema ou outras informações sensíveis.
7. Remover aplicações desnecessárias e documentação do sistema que possam revelar informações importantes para os atacantes.
8. Não incluir informações sensíveis nos parâmetros de requisição HTTP GET.
9. Desabilitar a funcionalidade de autocompletar nos formulários que contenham informações sensíveis, inclusive no formulário de autenticação.
10. Desabilitar o cache realizado 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 cabeçalhos HTTP “Pragma: no-cache”, que é menos efetivo, mas é compatível com HTTP/1.0.
11. A aplicação deve dar suporte a remoção de dados sensíveis quando os mesmos não forem mais necessários. Por exemplo: informação pessoal ou determinados dados financeiros.
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.

### Segurança nas comunicações:

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	12 de 20



**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

1. Utilizar criptografia na transmissão de todas as informações sensíveis. Isto deve incluir TLS para proteger a conexão e deve ser complementado por criptografia de arquivos que contém dados sensíveis ou conexões que não usam o protocolo HTTP.
2. Os certificados TLS devem ser válidos, possuir o nome de domínio correto, não estarem expirados e serem instalados com certificados intermediários, quando necessário.

Quando ocorre falha nas conexões TLS, o sistema não deve retornar uma conexão insegura.

1. Utilizar conexões TLS para todo conteúdo que requer acesso autenticado ou manutenção da
2. confidencialidade das informações sensíveis.
3. Utilizar um padrão único de implementação TLS que é configurado de modo apropriado.
4. Especificar a codificação dos caracteres para todas as conexões.
5. Filtrar os parâmetros que contenham informações sensíveis, provenientes do HTTP referer, quando realizar apontamentos para sites externos.

### Configuração do Sistema:

1. Garantir que os servidores, frameworks e componentes do sistema estão executando a última versão aprovada.
2. Garantir que os servidores, frameworks e componentes do sistema possuam os patches mais recentes aplicados para a versão em uso.
3. Desabilitar a listagem de diretórios.
4. Restringir os privilégios do servidor web, dos processos e das contas de serviços para o mínimo possível.
5. Quando exceções ocorrem no sistema, permitir que as falhas ocorram de modo seguro.
6. Remover todas as funcionalidades e arquivos desnecessários.
7. Remover o código de teste ou qualquer funcionalidade desnecessária para o ambiente de produção, antes que seja realizada a implantação do sistema.
8. Prevenir a divulgação da estrutura de diretórios impedindo que robôs de busca façam indexação de arquivos sensíveis, através da correta configuração do arquivo robots.txt, definindo diretórios que devem ser inacessíveis a estes indexadores em um diretório subjacente isolado. Assim, o acesso ao diretório pai definido no arquivo robots.txt deve estar desabilitado em vez de desabilitar cada diretório individualmente.
9. Definir quais métodos HTTP, Get ou Post, a aplicação irá suportar e se serão tratados de modo diferenciado nas diversas páginas da aplicação.
10. Desativar os métodos HTTP desnecessários, como extensões WebDAV. Caso for necessário o uso de algum método HTTP estendido para suportar manipulação de arquivos, então utilize algum mecanismo de autenticação bem controlado.
11. Se o servidor processa tanto requisições HTTP 1.0 e 1.1, certificar-se de que ambos são configurados de modo semelhante ou assegurar que qualquer diferença que possa existir seja compreendida (ex. manuseio de métodos HTTP estendidos).
12. Remover informações desnecessárias presentes nos cabeçalhos de resposta HTTP que podem estar relacionadas ao sistema operacional, versão do servidor web e frameworks de aplicação.

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	13 de 20





**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

13. O armazenamento da configuração de segurança para a aplicação deve ser capaz de ser produzida de forma legível para dar suporte à auditoria.
14. Implementar um sistema de gestão de ativos para manter o registro dos componentes e programas nele.
15. Isolar o ambiente de desenvolvimento da rede de produção e prover acesso somente para grupos de desenvolvimento e testes. Os ambientes de desenvolvimento comumente são configurados de modo menos seguro do que os ambientes de produção. Assim, os atacantes podem usar esse diferencial para descobrir vulnerabilidades compartilhadas ou encontrar caminhos para explorar as vulnerabilidades. Implementar um sistema de controle de mudanças para gerenciar e registrar as alterações no código, tanto do desenvolvimento, como dos sistemas em produção.

### Segurança em Base de Dados:

1. Usar consultas parametrizadas fortemente tipadas.
  2. Utilizar validação de entrada e codificação de saída e assegure a abordagem de meta caracteres. Se houver falha, o comando no banco de dados não deve ser executado.
  3. Certificar-se de que as variáveis são fortemente tipadas.
  4. Realizar a codificação (escaping) de meta caracteres em instruções SQL8.
  5. A aplicação deve usar o menor nível possível de privilégios ao acessar o banco de dados.
  6. Usar credenciais seguras para acessar o banco de dados.
  7. As strings de conexão não devem ser codificadas na aplicação. A string de conexão deve ser armazenada em um arquivo de configuração separado em um sistema confiável e as informações devem ser criptografadas.
  8. Usar procedimentos armazenados (stored procedures) para abstrair o acesso aos dados e permitir a remoção das permissões das tabelas no banco de dados.
  9. Encerrar a conexão tão logo seja possível.
  10. Remover ou modificar todas as senhas padrão de contas administrativas. Utilizar senhas robustas (incomuns ou difíceis de deduzir) ou implementar autenticação de múltiplos fatores. Desabilitar qualquer funcionalidade desnecessária no banco de dados, como stored procedures ou serviços desnecessários. Instale o mínimo conjunto de componentes ou opções necessárias (método de redução da área de superfície).
  11. Eliminar o conteúdo desnecessário incluído por padrão pelo fornecedor, ex: esquemas de exemplo.
  12. Desabilitar todas as contas criadas por padrão e que não são necessárias para suportar os requisitos de negócio.
  13. A aplicação deve se conectar ao banco de dados com diferentes credenciais de segurança para cada tipo de necessidade, como: usuário, somente leitura, convidado, administrador etc.
- Gerenciamento de Arquivos:
14. Não repassar dados fornecidos pelos usuários diretamente a uma função de inclusão dinâmica.
  15. Solicitar autenticação antes de permitir que seja feito o upload de um arquivo.
  16. Limitar os tipos de arquivos que podem ser enviados para aceitar somente os tipos que são

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	14 de 20



**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

17. necessários para os propósitos do negócio.
18. Validar se os arquivos enviados são do tipo esperado através da checagem dos cabeçalhos. Realizar a verificação de tipo de arquivo apenas pela extensão não é suficiente.
19. Não salvar arquivos no mesmo diretório de contexto da aplicação web. Os arquivos devem ser armazenados no servidor de conteúdo ou na base de dados.
20. Prevenir ou restringir upload de qualquer arquivo que possa ser interpretado pelo servidor web.
21. Desabilitar privilégios de execução nos diretórios de upload de arquivos.
22. Implementar o upload seguro nos ambientes UNIX por meio da montagem do diretório de destino como um unidade lógica, usando o caminho associado ou o ambiente de chroot.
23. No referenciamento de arquivos existentes, use uma lista branca (white list) de nomes e tipos de arquivos permitidos. Realize a validação do valor do parâmetro passado e caso não corresponda ao que é esperado, rejeite a entrada ou utilize um valor de arquivo especificado por padrão pela aplicação.
24. Não transmitir sem nenhum tratamento os dados informados pelo usuário a redirecionamentos dinâmicos. Se isto deve ser permitido, então o redirecionamento deve aceitar somente URLs relativas e validadas.
25. Não passar parâmetros de caminhos de diretórios ou arquivos nas requisições. Utilize algum mecanismo de mapeamento dos caminhos em disco para índices que são repassados para os usuários e servem para serem mapeados em uma lista pré-definida de caminhos dos arquivos.
26. Nunca enviar o caminho absoluto do arquivo para o cliente.
27. Certificar-se de que os arquivos da aplicação e os recursos são do tipo somente leitura.
28. Escanear arquivos que os usuários submeteram por mecanismo de upload em busca de vírus e malwares.

### Gerenciamento de Memória:

1. Utilize controle de entrada/saída para dados não confiáveis.
2. Verificar se o buffer é tão grande quanto o especificado.
3. Ao usar funções que aceitam um determinado número de bytes para realizar cópias, como
4. strncpy(), esteja ciente de que se o tamanho do buffer de destino for igual ao tamanho do buffer de origem, ele não pode encerrar a sequência de caracteres com valor nulo (null).
5. Verificar os limites do buffer caso as chamadas a função são realizadas em um loop e verificar se não há nenhum perigo de escrever além do espaço alocado.
6. Truncar todas as strings de entrada para um tamanho razoável antes de passá-las para as funções de cópia e concatenação.
7. Encerre os recursos de modo específico, sem contar com o garbage collector na liberação dos recursos alocados para objetos de conexão, identificadores de arquivo etc.
8. Usar pilhas não-executáveis, quando disponíveis.
9. Evitar o uso de funções reconhecidas por serem vulneráveis, por exemplo: printf, strcat, strcpy, etc.
10. Liberar a memória alocada de modo apropriado após concluir a sub-rotina (função/método) e em todos os pontos de saída.

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	15 de 20





**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

### Práticas Gerais de Codificação:

1. Utilizar sempre código testado, gerenciado e aprovado em vez de criar código novo, não gerenciado, para tarefas comuns.
2. Utilizar APIs que embutem tarefas específicas para realizar tarefas do sistema operacional. Não permitir que a aplicação execute comandos diretamente no sistema operacional, especialmente através da utilização de shells de comando iniciados pela aplicação.
3. Utilize mecanismo de verificação de integridade por checksum ou hash para verificar a integridade do código interpretado, bibliotecas, arquivos executáveis e arquivos de configuração.
4. Utilize mecanismos de lock para evitar requisições simultâneas para a aplicação ou utilize um mecanismo de sincronização para evitar condições de disputa (race conditions).
5. Proteja as variáveis compartilhadas e recursos contra acessos concorrentes inapropriados.
6. Inicialize explicitamente todas as variáveis e outros dados persistidos, durante a declaração ou antes do primeiro uso da variável.
7. Nos casos em que a aplicação deve ser executada com privilégios elevados, aumente os privilégios o mais tarde possível e revogue os privilégios tão logo seja possível.  
Evitar erros de cálculo decorrentes da falta de entendimento da representação interna da linguagem de programação usada e como é realizada a interação com os aspectos de cálculo numérico. Assim, deve-se prestar bastante atenção para as discrepâncias de tamanho de byte, precisão, distinções de sinal (signed/unsigned), truncamento, conversão e casting entre os tipos, cálculos que devolvem erros do tipo "not-a-number", e como a linguagem de programação trata números muito grandes ou muito pequenos para sua representação interna.
8. Não repassar diretamente dados fornecidos pelo usuário para qualquer função de execução dinâmica, sem antes realizar o tratamento dos dados de modo adequado.
9. Restringir os usuários de gerar um novo código ou alterar o código existente.
10. Revisar 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.
11. Implementar atualizações de modo seguro. Se a aplicação deve realizar atualizações automáticas, então utilize mecanismos de assinatura digital para garantir a integridade do código e garanta que os clientes façam a verificação da assinatura após o download. Use canais criptografados para transferir o código a partir do host do servidor.

## 7. CONTROLE DE REGISTROS

Identificação	Armazenamento e Preservação	Proteção (acesso)	Recuperação e uso	Retenção	Disposição
Norma de SI	Eletrônico (GED)	Restrição de acesso	Por tema ou título	Até a próxima atualização	Substituição

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	16 de 20

**Uso Interno**

Tipo de Documento: **Procedimento**  
 Área de Aplicação: **Segurança da Informação**  
 Título do Documento: **Checklist de Práticas de Codificação Segura**

				do documento	
--	--	--	--	--------------	--

## 8. ANEXOS

### ANEXO I (Glóssário)

#### ANEXO I - (Glossário)

**Agente de Ameaça:** Qualquer entidade que pode ter um impacto negativo sobre o sistema. Este pode ser tanto um usuário mal-intencionado que quer comprometer os controles de segurança do sistema, como um

desvio accidental do sistema ou uma ameaça física como incêndios ou inundações.

**Autenticação:** É um conjunto de controles que são usados para verificar a identidade de um usuário, ou

outra entidade, que interage com o software.

**Autenticação Seqüencial:** Ocorre quando os dados de autenticação são solicitados em sucessivas páginas,

em vez de serem solicitados em uma única página.

**Autenticação de múltiplos fatores:** É um processo de autenticação que requer vários tipos de credenciais

produzidas pelo usuário. Normalmente é baseado em algo que eles possuem (ex.: cartão inteligente), algo

que eles sabem (uma ex.: senha), ou algo que eles são (ex.: dados provenientes de um leitor biométrico).

**Canonicalização:** É uma operação realizada para reduzir várias codificações e representações de dados em

uma única forma simplificada.

**Caracteres Maliciosos:** São caracteres ou representação codificada de um caractere que podem produzir

efeitos indesejáveis sobre a operação normal da aplicação ou dos sistemas associados quando são interpretados, por terem significado especial. Estes caracteres podem ser usados para:

- Modificar a estrutura de código existente
- Inserir novo código não intencional
- Modificar caminhos
- Causar saídas inesperadas das funções ou rotinas
- Causar condições de erro
- Ter qualquer dos efeitos anteriores nas aplicações subjacentes

**Casos de Abuso:** Descreve o mau uso intencional (ou não) do software. Os casos de abuso desafiam os

pressupostos do projeto do sistema.

**Codificação de entidade HTML:** Compreende ao processo de substituição de determinados caracteres

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	17 de 20



**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

ASCII com seus equivalentes em entidade HTML. Por exemplo, a codificação poderia substituir o caractere “<” com o equivalente em HTML “&lt;”. As entidades HTML são inertes na maioria dos interpretadores, especialmente navegadores, que podem atenuar os ataques do lado do cliente.

Codificação de saída baseada em contexto: Ocorre quando a codificação de dados de saída é realizada com base em como ela será utilizada pela aplicação. Os métodos específicos variam dependendo da forma como os dados são usados. Se os dados de saída estiverem incluídos na resposta ao cliente, é levado em conta como cenários de inclusão, situações como: o corpo de um documento HTML, um atributo de HTML, codificação JavaScript, codificação CSS ou em uma URL. Deve também ser levado em conta outros casos como consultas SQL, XML e LDAP.

Codificação de saída de dados: É um conjunto de controles que abordam o uso de codificação para garantir que a saída de dados gerada pela aplicação seja segura.

Confidencialidade: Para garantir que as informações sejam divulgadas apenas para as partes autorizadas.

Configuração do Sistema: É um conjunto de controles que ajudam a garantir se os componentes de infraestrutura de apoio ao software são implementados de forma segura.

Consultas parametrizadas (prepared statements): Mantém a consulta e os dados separados através do

uso de espaços reservados. A estrutura de consulta é definida com caracteres especiais que representam os

parâmetros da consulta a serem substituídos. A instrução SQL é enviada para o banco de dados e preparada

para receber os parâmetros. Em seguida, a consulta parametrizada (prepared statement) é combinada com

os valores dos parâmetros. Isto impede que a consulta seja alterada, pois, os valores dos parâmetros são

combinados com a declaração compilada, e não concatenados diretamente a uma sequência SQL.

Controle de Acesso: É um conjunto de controles que libera ou nega o acesso a um recurso do sistema a um

usuário, ou qualquer outra entidade. Normalmente é baseado em regras hierárquicas e privilégios individuais associados a papéis, porém também inclui interações entre sistemas.

Controles de Segurança: Uma ação que elimina uma vulnerabilidade potencial e ajuda a garantir que o software se comporte conforme o esperado.

Cross Site Request Forgery (CSRF): Uma aplicação ou site web externos forçam o navegador do cliente

a realizar uma requisição involuntária para um aplicativo que o cliente possua uma sessão ativa. As aplicações são vulneráveis quando usam URLs e parâmetros conhecidas ou previsíveis, ou quando o

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	18 de 20



**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura

navegador automaticamente transmite todas as informações solicitadas pela sessão em cada solicitação para a aplicação vulnerável. Isto é apenas um dos ataques discutidos de modo específico neste documento e só

está incluído porque a vulnerabilidade associada é muito comum e mal compreendida.

Disponibilidade: Propriedade de estar acessível e utilizável quando demandado por uma entidade autorizada.

Estado dos Dados: Ocorre quando os dados ou parâmetros são usados, através da aplicação ou servidor, para emular uma conexão persistente ou controlar o status de um cliente através de um processo de múltiplas requisições ou transações.

Exploit: É uma ação que tira proveito de uma vulnerabilidade. Normalmente é uma ação intencional destinada a comprometer os controles de segurança do software, aproveitando a existência de uma vulnerabilidade.

Gerência de Arquivo: É um conjunto de controles que abrangem a interação entre o código da aplicação e os arquivos do sistema.

Gerenciamento de memória: É um conjunto de controles que tratam do uso de memória e do buffer.

Gerenciamento de Sessão: É um conjunto de controles que ajudam a garantir que as aplicações web lidem com sessões http de um modo seguro.

Tratamento de erros e log: É um conjunto de práticas que garantem que a aplicação realiza o tratamento dos erros de modo seguro, como também realiza de modo apropriado o registro de log dos eventos. Tratamento dos Dados: É o processo de tornar seguros os dados potencialmente prejudiciais através do processo de remoção, substituição, codificação ou escaping dos caracteres. Validação de Entrada de Dados: É o conjunto de controles que verificam se as propriedades de todas as entradas de dados correspondem ao que é esperado pela aplicação, como tipo dos dados, tamanho, intervalos, conjunto de caracteres aceitáveis que não contenham caracteres maliciosos. Vulnerabilidade: É uma fragilidade que torna o sistema suscetível a um ataque ou danos.

## 9. REGISTRO DE ALTERAÇÕES

### 9.1. Colaboradores

Empresa	Área	Nome
CPFL Energia	EIS	Grasiela Castro
CPFL Energia	EIS	Mateus Rocha

### 9.2. Alterações

Versão Anterior	Data da Versão Anterior	Alterações em relação à Versão Anterior
N/A	N/A	Documento em versão inicial

Nº Documento:	Categoria:	Versão:	Aprovado por:	Data Publicação:	Página:
19280	Instrução	1.0	Emerson Cardoso	27/07/2022	19 de 20



**Uso Interno**

Tipo de Documento:	Procedimento
Área de Aplicação:	Segurança da Informação
Título do Documento:	Checklist de Práticas de Codificação Segura