Procedimento para Desenvolvimento Seguro de Aplicações Web

Código: **PR_27**
Versão: **1.0**

Título:	
Procedimento para Desenvolvimento Seguro de Aplicações Web	
Diretoria:	Gerência Responsável:
DSC/STI	**Gerência de Tecnologia do Negócio**
Dono do Processo:	Elaboradores:
Fábio Leandro Bernardes Duarte	**Rogério Oliveira Ferreira** **Darley Fernandes da Silva**

Controle de Versão

Histórico (para uso da Gerência de Governança):

Versão	Data Publicação	Elaborado	Motivo da Versão
1.0	01/02/2017	Gerência de Infraestrutura	Tornar as aplicações web mais seguras
2.0	22/07/2022	Rogério Ferreira	Atualização e recomendações de práticas de segurança da informação com vistas a mitigar riscos de vulnerabilidades em aplicações web.

Aprovação da Versão Atual:

Nome	Cargo
Fábio Leandro Bernardes Duarte	Gerente de Tecnologia do Negócio

Título:	
Procedimento para Desenvolvimento Seguro de Aplicações Web	
Diretoria:	Gerência Responsável:
DSC/STI	**Gerência de Tecnologia do Negócio**
Dono do Processo:	Elaboradores:
Fábio Leandro Bernardes Duarte	**Rogério Oliveira Ferreira** **Darley Fernandes da Silva**

Sumário

Título:	
Procedimento para Desenvolvimento Seguro de Aplicações Web	
Diretoria:	Gerência Responsável:
DSC/STI	**Gerência de Tecnologia do Negócio**
Dono do Processo:	Elaboradores:
Fábio Leandro Bernardes Duarte	**Rogério Oliveira Ferreira** **Darley Fernandes da Silva**

1. Justificativa

O desenvolvimento seguro de aplicações web é primordial para que as organizações não venham a sofrer impactos importantes provenientes de ataques cibernéticos pela exploração de vulnerabilidade em aplicação web. Segurança cibernética tem sido uma preocupação constante nas organizações devido ao alto índice de ameaças que exploram vulnerabilidades em seus ambientes tecnológicos. A busca por melhores práticas em matéria de segurança em aplicações web tem levado a implementação de algumas práticas baseadas em frameworks como Open Web Application Security Project (OWASP) e do Center for Internet Security (CIS).

2. Objetivo

O objetivo deste procedimento é auxiliar os profissionais de desenvolvimento e manutenção de sistemas da Superintendência de Tecnologia da Informação - STI a adotarem práticas recomendadas de segurança da informação com vistas a mitigar riscos de vulnerabilidades em aplicações web.

3. Escopo

O escopo deste documento abrange todas as aplicações web a serem desenvolvidas, seja por profissionais da STI ou fornecedores contratados.

4. Referências

OWASP - Open Web Application Security Project
CIS - Center for Internet Security

5. Considerações Gerais

A equipe da STI de infraestrutura e desenvolvimento é responsável por validar todos os requisitos constantes neste documento para que aplicações web não entrem em ambiente de produção com vulnerabilidades que de alguma forma possam incidir nos casos de ataques apresentados abaixo, mas não se limitando a estes:

Casos de abuso	Riscos
Injeção de código	Ocorrem quando dados não confiáveis são enviados para um interpretador como parte de um comando ou consulta. Os dados manipulados pelo atacante podem iludir o interpretador para que este execute comandos indesejados ou permita o acesso a dados não autorizados.

Casos de abuso	Riscos
Quebra de Autenticação e Gerenciamento de Sessão	Ocorre quando as funções da aplicação relacionadas à autenticação e gerenciamento de sessão são implementadas de forma incorreta, permitindo que os atacantes comprometam senhas, chaves e tokens de sessão ou explorem outra falha da implantação para assumir a identidade de outros usuários.
Cross-Site Scripting (XSS)	Falhas XSS permitem aos atacantes executarem *scripts* no navegador da vítima.
Referência Insegura e Direta a Objetos	Ocorre quando um programador expõe uma referência à implantação interna de um objeto, como um arquivo, diretório, ou registro da base de dados.
Exposição de Dados Sensíveis	Ocorre quando as aplicações não protegem devidamente os dados sensíveis, tais como cartões de crédito, IDs fiscais e credenciais de autenticação.
Falta de Função para Controle do Nível de Acesso	Ocorre quando as aplicações não verificam os direitos de acesso, em nível de função, antes de tornar essa funcionalidade visível na interface do usuário.
Cross-Site Request Forgery (CSRF)	Ocorre quando o navegador da vítima é forçado a executar uma ação maliciosa em favor do atacante.
Utilização de Componentes Vulneráveis Conhecidos	Componentes, tais como bibliotecas, *frameworks*, e outros módulos de *software* quase sempre são executados com privilégios elevados.
Redirecionamentos e Encaminhamentos Inválidos	Aplicações frequentemente redirecionam e encaminham usuários para outras páginas ou sites, e usam dados não confiáveis para determinar as páginas de destino.

6. Descrição do Procedimento

Nesta sessão do documento estão descritos os requisitos para desenvolvimento seguro bem como o detalhamento dos controles de segurança críticos que devem ser observados para o desenvolvimento seguro:

6.1 Requisito 1: Validação dos dados de entrada

As vulnerabilidades baseadas nos dados de entrada podem surgir em qualquer funcionalidade de uma aplicação, e podem estar presentes em praticamente toda tecnologia de uso comum existente. Uma grande variedade de ataques a aplicações web se origina ou envolve o envio de dados de entrada

inesperados ou especialmente construídos para causar comportamentos inesperados na aplicação. Sendo assim, toda entrada de dados em um sistema deve ser considerada, a princípio, não confiável.

Subcontroles de Segurança Críticos - Validação dos dados de entrada
1.1 Efetuar toda a validação dos dados em um sistema confiável – por exemplo, centralizar todo o processo no servidor.
1.2 Identificar todas as fontes de dados e classificá-las como sendo confiáveis ou não. Em seguida, validar os dados provenientes de fontes nas quais não se possa confiar (ex: base de dados, stream de arquivos etc.).
1.3 A rotina de validação de dados de entrada deve ser centralizada na aplicação.
1.4 Especificar conjunto de caracteres apropriado, como UTF-8, para todas as fontes de entrada de dados.
1.5 Codificar os dados para um conjunto de caracteres comuns antes da validação (Canonicalize).
1.6 Quando há falha de validação, a aplicação deve rejeitar os dados fornecidos.
1.7 Determinar se o sistema suporta conjuntos de caracteres estendidos UTF-8 e, em caso afirmativo, validar após efetuar a descodificação UTF-8.
1.8 Validar todos os dados provenientes dos clientes antes do processamento, incluindo todos os parâmetros, campos de formulário, conteúdo das URLs e cabeçalhos HTTP, como, por exemplo, os nomes e os valores dos Cookies. Certificar-se, também, de incluir mecanismos automáticos de postback nos blocos de código JavaScript, Flash ou qualquer outro código embutido.
1.9 Verificar se os valores de cabeçalho, tanto das requisições, como das respostas, contêm apenas caracteres ASCII.
1.10 Validar dados provenientes de redirecionamentos. Os atacantes podem incluir conteúdo malicioso diretamente para o alvo do mecanismo de redirecionamento, podendo assim contornar a lógica da aplicação e qualquer validação executada antes do redirecionamento.
1.11 Validar tipos de dados esperados.
1.12 Validar intervalo de dados.
1.13 Validar o tamanho dos dados.
1.14 Validar, sempre que possível, todos os dados de entrada através de um método baseado em “listas de permissões" (whitelists) que utilizem uma lista de caracteres ou expressões regulares para definirem os caracteres permitidos.
1.15 Se qualquer caractere potencialmente perigoso precisa ser permitido na entrada de dados da aplicação, certificar-se de que foram implementados controles adicionais 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: <, >, ", ', %, (,), &, +, \, \', \".
1.16 Se a rotina de validação padrão não abordar as seguintes entradas, então elas devem ser verificadas:
a) Verificar bytes nulos (%00)
b) Verificar se há caracteres de nova linha (%0d, %0a, \r, \n)
c) Verificar se há caracteres “ponto-ponto barra” (../ ou ..\) que alteram caminhos. Nos casos de conjunto de caracteres que usem a extensão UTF-8, o sistema deve utilizar casos de conjunto de caracteres que usem a extensão UTF-8, o sistema deve utilizar representações alternativas como: %c0%ae%c0%ae/. A canonicalização deve ser utilizada para resolver problemas de codificação dupla (double encoding) ou outras formas de ataques por ofuscação

6.2 Requisito 2: Codificação de dados de saída

Com a diversidade de arquiteturas modernas de aplicações web, a realização da codificação de saída é muito importante. Pode ser difícil fornecer validação de entrada robusta em certos cenários. Portanto, o uso de APIs mais seguras com consultas parametrizadas, estruturas de modelagem com escape

automático ou codificação de saída cuidadosamente escolhida é fundamental para a segurança da aplicação.

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

Subcontroles de Segurança Críticos - Codificação de dados de saída
2.1 Efetuar toda a codificação dos dados em um sistema confiável - por exemplo, centralizar todo o processo no servidor.
2.2 Utilizar uma rotina padrão e testada para cada tipo de codificação de saída.
2.3 Realizar a codificação, baseada em contexto, de todos os dados enviados para o cliente que têm origem em um ambiente fora dos limites de confiança da aplicação. A codificação das entidades HTML é um exemplo, mas nem sempre funciona para todos os casos.
2.4 Codificar todos os caracteres, a menos que sejam conhecidos por serem seguros para o interpretador de destino.
2.5 Realizar o tratamento (sanitização), baseado em contexto, de todos os dados provenientes de fontes não confiáveis usados para construir consultas SQL, XML e LDAP.
2.6 Tratar todos os dados provenientes de fontes que não sejam confiáveis que gerem comandos para o sistema operacional.

6.3 Requisito 3: Autenticação e gerenciamento de credenciais

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

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

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

Subcontroles de Segurança Críticos - Autenticação e gerenciamento de credenciais
3.1 Requerer autenticação para todas as páginas e recursos, exceto para aqueles que são intencionalmente públicos.
3.2 Os controles de autenticação devem ser executados em um sistema confiável - por exemplo, centralizar todo o processo no servidor.
3.3 Sempre que possível, estabelecer e utilizar serviços de autenticação padronizados e testados.
3.4Utilizar uma implementação centralizada para realizar os procedimentos de autenticação, disponibilizando bibliotecas que invoquem os serviços externos de autenticação.
3.5 Separar a lógica de autenticação do recurso que está a ser requisitado e usar redirecionadores dos controladores de autenticação centralizados.

Subcontroles de Segurança Críticos - Autenticação e gerenciamento de credenciais
3.6 Quando ocorrerem situações excepcionais nos controles de autenticação, executar procedimentos em caso de falha com o propósito de manter o sistema seguro.
3.7 Todas as funções administrativas e de gerenciamento de contas devem ser tão seguras quanto o mecanismo de autenticação principal.
3.8 Se a aplicação gerenciar um repositório de credenciais, esta deverá garantir que as senhas são armazenadas na base de dados somente sob a forma de resumo/hash da senha na forma de one-way salted hashes e que a tabela/arquivo que armazena as senhas e as próprias chaves são manipuladas apenas pela aplicação. Não utilizar algoritmos de hash reconhecidamente inseguros.
3.9 A geração dos resumos (hash) das senhas deve ser executada em um sistema confiável - por exemplo, centralizar o controle no servidor.
3.10 Validar os dados de autenticação somente no final de todas as entradas de dados, especialmente para as implementações de autenticação sequencial.
3.11 As mensagens de falha na autenticação não devem indicar qual parte dos dados de autenticação está incorreta. Por exemplo, em vez de exibir mensagens como “Nome de usuário incorreto” ou “Senha incorreta”, utilize apenas mensagens como: “Usuário e/ou senha inválidos”, para ambos os casos de erro. As respostas de erro devem ser idênticas nos dois casos.
3.12 Utilizar autenticação para conexão a sistemas externos que envolvam tráfego de informação sensível ou acesso às funções.
3.13 As credenciais de autenticação para acessar serviços externos à aplicação devem ser cifradas e armazenadas em um local protegido de um sistema confiável - por exemplo, no servidor da aplicação. Obs.: o código-fonte não é considerado um local seguro.
3.14 Utilizar apenas requisições POST para transmitir credenciais de autenticação.
3.15 Somente trafegar senhas (não temporárias) através de uma conexão protegida (SSL/TLS) ou no formato de dado cifrado, como no caso de envio de e-mail cifrado. Senhas temporárias enviadas por e-mail podem ser um caso de exceção aceitável.
3.16 Exigir que os requisitos de complexidade de senha estabelecidos pela política ou regulamento sejam cumpridos. As credenciais de autenticação devem resistir a ataques que, tipicamente, ameaçam o ambiente de produção. Um exemplo pode ser a exigência do uso simultâneo de caracteres alfabéticos, numéricos e/ou caracteres especiais.
3.17 Exigir que os requisitos de comprimento de senha estabelecidos pela política ou pelo regulamento sejam cumpridos. O uso de oito caracteres é o mais comum, porém usar 8 é mais recomendado. Considere, ainda, o uso de senhas que contenham várias palavras (uma frase).
3.18 A entrada da senha deve ser ocultada na tela do usuário. Em HTML, utilizar o campo do tipo "password".
3.19 Desativar a conta após um número pré-definido de tentativas inválidas de autenticação (e.g. cinco tentativas é o mais comum). A conta deve ser desativada por um período suficientemente longo para desencorajar a dedução das credenciais pelo método de força bruta, mas não tão longo ao ponto de permitir um ataque de negação de serviço.
3.20 Os processos de redefinição de senhas e operações de mudanças devem exigir os mesmos níveis de controle previstos para a criação de contas e autenticação
3.21 Esquemas de pergunta/resposta (pré-definidos) usados para a redefinição da senha devem evitar ataques que lancem respostas aleatórias. Por exemplo, “livro favorito” é uma questão fraca, pois “A Bíblia” é uma resposta muito comum.
3.22 Se optar por usar redefinição de senha baseada em e-mail, envie um e-mail somente para o endereço pré-definido contendo um link ou uma senha de acesso temporário que permitam ao usuário redefinir a senha.

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

6.4 Requisito 4: Gerenciamento de sessões

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

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

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

Subcontroles de Segurança Críticos - Gerenciamento de sessões
4.1 Utilizar controles de gerenciamento de sessão baseados no servidor ou em framework. A aplicação deve reconhecer apenas esses identificadores de sessão como válidos.
4.2 A criação dos identificadores de sessão deve ser sempre realizada em um sistema confiável - por exemplo, centralizado no servidor.
4.3 O controle de gestão de sessão deve usar algoritmos conhecidos, padronizados e bem testados que garantam a aleatoriedade dos identificadores de sessão.

Subcontroles de Segurança Críticos - Gerenciamento de sessões
4.4 Definir o domínio e o caminho para os cookies que contenham identificadores de sessão autenticados, para um valor devidamente restrito ao site.
4.5 A funcionalidade de saída (logout) deve encerrar completamente a sessão ou conexão associada.
4.6 A funcionalidade de saída (logout) deve estar disponível em todas as páginas que requerem autenticação.
4.7 Estabelecer um tempo de expiração da sessão que seja o mais curto possível, baseado no balanceamento dos riscos e requisitos funcionais do negócio. Na maioria dos casos, não deve ser maior que algumas horas.
4.8 Não permitir logins persistentes (sem prazo de expiração), e realizar o encerramento da sessão periodicamente, mesmo quando ela estiver ativa. Isso deve ser feito, especialmente, em aplicações que suportam várias conexões de rede ou que se conectam a sistemas críticos. O tempo de encerramento deve estar em sintonia com os requisitos do negócio e o usuário deve receber notificações suficientes para atenuar os impactos negativos dessa medida.
4.9 Se uma sessão estava estabelecida antes do login, ela deve ser encerrada para que uma nova seja estabelecida após o login.
4.10 Gerar um novo identificador de sessão quando houver uma nova autenticação.
4.11 Não permitir conexões simultâneas com o mesmo identificador de usuário.
4.12 Não expor os identificadores de sessão em URLs, mensagens de erro ou logs. Os identificadores de sessão devem apenas ser encontrados no cabeçalho do cookie HTTP. Por exemplo, não trafegar os identificadores de sessão sob a forma de parâmetros GET.
4.13 Proteger os dados de sessão do lado servidor contra acessos não autorizados por outros usuários do servidor, através da implementação de controles de acesso apropriados no servidor.
4.14 Gerar um novo identificador de sessão e desativar o antigo periodicamente. Isso pode mitigar certos cenários de ataques de sequestro de sessão (session hijacking), quando o identificador de sessão original for comprometido.
4.15 Gerar um novo identificador de sessão caso a segurança da conexão mude de HTTP para HTTPS, como pode ocorrer durante a autenticação. Internamente à aplicação, é recomendável utilizar HTTPS de forma constante em vez de alternar entre HTTP e HTTPS.
4.16 Utilizar mecanismos complementares ao mecanismo padrão de gerenciamento de sessões para operações sensíveis do lado servidor, como no caso de operações de gerenciamento de contas, através da utilização de tokens aleatórios ou parâmetros associados à sessão. Esse método pode ser usado para prevenir ataques do tipo Cross Site Request Forgery (CSRF).
4.17 Utilizar mecanismos complementares ao gerenciamento de sessões para operações altamente sensíveis ou críticas, utilizando tokens aleatórios ou parâmetros em cada requisição em vez de basear-se apenas na sessão.
4.18 Configurar o atributo "secure" para cookies transmitidos através de uma conexão TLS.
4.19 Configurar os cookies com o atributo “HttpOnly”, a menos que seja explicitamente necessário ler ou definir os valores deles através de scripts do lado cliente da aplicação.

6.5 Requisito 5: Controle de acesso

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

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

Devido à natureza complexa dos requisitos típicos de controle de acesso, tal função é uma fonte frequente de vulnerabilidades de segurança que permitem a um invasor obter acesso não autorizado a dados e funcionalidades.

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

Devido à prevalência de falhas de controle de acesso, no entanto, esse esforço é uma atividade que pode fornecer benefícios indevidos a um autor de ataque a uma aplicação da web.

Vide abaixo uma lista de subcontroles a serem considerados nessa hipótese.

Subcontroles de Segurança Críticos - Controle de acesso
5.1 Utilizar apenas objetos do sistema que sejam confiáveis, como ocorre com os objetos de sessão do servidor, para realizar a tomada de decisão sobre a autorização de acesso.
5.2 Utilizar um único componente em toda a aplicação Web para realizar o processo de verificação de autorização de acesso. Isto inclui bibliotecas que invocam os serviços externos de autorização.
5.3 Quando ocorrer alguma falha no controle de acesso, ela deve ocorrer de modo seguro, sem que sejam repassados detalhes que possam facilitar ataques direcionados.
5.4 Negar todos os acessos, caso a aplicação não consiga ter acesso às informações contidas na configuração de segurança.
5.5 Garantir o controle de autorização em todas as requisições, inclusive em scripts do lado servidor, "includes" e requisições provenientes de tecnologias do lado cliente.
5.6 Isolar do código da aplicação os trechos de código que contêm lógica privilegiada.
5.7 Restringir o acesso aos arquivos e outros recursos, incluindo aqueles que estão fora do controle direto da aplicação, somente aos usuários autorizados.
5.8 Restringir o acesso às URLs protegidas somente aos usuários autorizados.
5.9 Restringir o acesso às funções protegidas somente aos usuários autorizados.
5.10 Restringir o acesso às referências diretas aos objetos somente aos usuários autorizados.
5.11 Restringir o acesso aos serviços somente aos usuários autorizados.
5.12 Restringir o acesso aos dados da aplicação somente aos usuários autorizados.
5.13 Restringir o acesso aos atributos e dados dos usuários, bem como informações das políticas usadas pelos mecanismos de controle de acesso.
5.14 Restringir o acesso às configurações de segurança relevantes apenas aos usuários autorizados.
5.15 As regras de controle de acesso representadas pela camada de apresentação devem coincidir com as regras presentes no lado servidor.
5.16 Se o estado dos dados deve ser armazenado no lado cliente, utilizar mecanismos de criptografia e verificação de integridade no lado servidor para detectar possíveis adulterações.
5.17 Garantir que os fluxos lógicos da aplicação obedecem às regras de negócio.
5.18 Limitar o número de transações que um único usuário ou dispositivo pode executar em determinado período. As transações por período devem estar acima das necessidades reais do negócio, mas abaixo o suficiente para impedir ataques automatizados.
5.19 Utilizar o campo “referer” do cabeçalho somente como forma de verificação suplementar. Ele não deve ser usado sozinho como forma de validação de autorização, pois pode ter o valor adulterado.

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

6.6 Requisito 6: Criptografia

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

Alguns requisitos de alto nível devem ser observados:

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

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

Subcontroles de Segurança Críticos - Criptografia
6.1 Todas as funções de criptografia utilizadas para proteger dados sensíveis dos usuários da aplicação devem ser implantadas em um sistema confiável - neste caso, o servidor.
6.2 A senha mestra deve ser protegida contra acessos não autorizados.
6.3 Quando ocorrer alguma falha nos módulos de criptografia, permitir que ela ocorra de modo seguro.

Subcontroles de Segurança Críticos - Criptografia
6.4 Todos os números, nomes de arquivos, GUIDs e strings aleatórias devem ser gerados usando um módulo criptográfico com gerador de números aleatórios, somente se os valores aleatórios gerados forem impossíveis de serem deduzidos.
6.5 Os módulos de criptografia usados pela aplicação devem ser compatíveis com a FIPS 140-2 ou com um padrão equivalente (http://csrc.nist.gov/groups/STM/cmvp/validation.html).
6.6 Estabelecer e utilizar uma política e um processo que defina como é realizado o gerenciamento das chaves criptográficas.

6.7 Requisito 7: Tratamento de erros e logs

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

Logs com informação de qualidade normalmente possuem dados sensíveis e devem ser protegidos conforme a Norma Complementar nº 21 IN01/DSIC/GSIPR, que trata de diretrizes para o Registro de Eventos, Coleta e Preservação de Evidências de Incidentes de Segurança. Por geralmente possuírem informações bastante sensíveis, os logs também podem ser um alvo bem atraente para os atacantes.

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

Subcontroles de Segurança Críticos - Tratamento de erros e logs
7.1 Não expor informações sensíveis nas repostas de erros, inclusive detalhes de sistema, identificadores de sessão ou informação da conta do usuário.
7.2 Usar mecanismos de tratamento de erros que não mostrem informações de depuração (debug) ou informações da pilha de exceção.
7.3 Usar mensagens de erro genéricas e páginas de erro personalizadas.
7.4 A aplicação deve tratar os erros sem se basear nas configurações do servidor.
7.5 A memória alocada deve ser liberada de modo apropriado quando ocorrerem condições de erro.
7.6 O tratamento de erros lógicos associados com os controles de segurança deve, por padrão, negar o acesso.
7.7 Todos os controles de log devem ser implementados em um sistema confiável, por exemplo, centralizar todo o processo no servidor.
7.8 Os controles de log devem dar suporte tanto para os casos de sucesso como os de falha relacionados com os eventos de segurança.
7.9 Garantir que os logs armazenam eventos importantes.
7.10 Garantir que as entradas de log que incluam dados nos quais não se confia não sejam executadas como código-fonte na interface de visualização de logs.
7.11 Restringir o acesso aos logs apenas para pessoal autorizado.
7.12 Utilizar uma rotina centralizada para realizar todas as operações de log.
7.13 Não armazenar informações sensíveis nos registros de logs, como detalhes desnecessários do sistema, identificadores de sessão e senhas
7.15 Garantir o uso de algum mecanismo que conduza (ou facilite) o processo de análise de logs.

Subcontroles de Segurança Críticos - Tratamento de erros e logs
7.16 Colete logs do provedor de serviços, onde houver suporte. Exemplos de implementações incluem coleta de eventos de autenticação e autorização, eventos de criação e de descarte de dados e eventos de gestão de usuários.

6.8 Requisito 8: Proteção de dados

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

Subcontroles de Segurança Críticos - Proteção de dados
8.1 Implementar uma política de privilégio mínimo, restringindo aos usuários apenas às funcionalidades, dados e informações do sistema que são necessárias para executarem suas tarefas.
8.2 Proteger contra acesso não autorizado todas as cópias temporárias ou registradas em cache que contenham dados sensíveis e estejam armazenadas no servidor; excluir esses arquivos logo que não forem mais necessários.
8.3 Criptografar informações altamente sensíveis quando armazenadas – como dados de verificação de autenticação – mesmo que estejam no lado servidor, usando sempre algoritmos conhecidos, padronizados e bem testados. Consulte a seção que trata sobre “Práticas de Criptografia” para orientações adicionais.
8.4 Proteger o código-fonte presente no servidor para que não seja acessado por usuários não autorizados.
8.5 Não armazenar senhas, strings de conexão ou outras informações confidenciais em texto claro/legível ou em qualquer forma criptograficamente insegura no lado cliente. Isso é válido também quando há utilização de formatos inseguros, como MS viewstate ou código compilado que é executado no lado cliente.
8.6 Remover comentários do código de produção que podem ser acessados pelos usuários e podem revelar detalhes internos do sistema ou outras informações sensíveis.
8.7 Remover aplicações desnecessárias e documentação do sistema que possam revelar informações importantes para os autores de ataques.
8.8 Não incluir informações sensíveis nos parâmetros de requisição HTTP GET.
8.9 Desativar a funcionalidade de autocompletar nos formulários que contenham informações sensíveis, inclusive no formulário de autenticação.
8.10 Desativar a cache realizada no lado cliente das páginas que contenham informações sensíveis. O parâmetro "Cache-Control: no-store" pode ser usado em conjunto com o controle definido no cabeçalho HTTP “Pragma: no-cache”, que é menos efetivo, porém compatível com HTTP/1.0 .
8.11 A aplicação deve dar suporte à remoção de dados sensíveis quando estes não forem mais necessários - como, por exemplo, informação pessoal ou dados financeiros.
8.12 Implementar mecanismos de controle de acesso apropriados para dados sensíveis armazenados no servidor. Isto inclui dados em cache, arquivos temporários e dados que devem ser acessíveis somente por usuários específicos do sistema.
8.13 Prover mecanismos que garantam a anonimização dos dados pessoais e dados pessoais sensíveis, conforme a Lei Geral de Proteção de Dados Pessoais brasileira.

6.9 Requisito 9: Segurança nas comunicações

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

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

Subcontroles de Segurança Críticos - Segurança nas comunicações
9.1 Utilizar criptografia na transmissão de todas as informações sensíveis. Isto deve incluir TLS para proteger a conexão e deve ser complementado com criptografia de arquivos que contém dados sensíveis ou conexões que não usam o protocolo HTTP.
9.2 Os certificados TLS devem ser válidos, possuir o nome de domínio correto, não estar expirados e ser instalados com certificados intermediários, quando necessário.
9.3 Quando ocorrer alguma falha nas conexões TLS, o sistema não deve fornecer uma conexão insegura.
9.4 Utilizar conexões TLS para todo o conteúdo que requerer acesso autenticado ou que contenha informação sensível.
9.5 Utilizar TLS para conexões com sistemas externos que envolvam funções ou informações sensíveis.
9.6 Utilizar um padrão único de implementação TLS configurado de modo apropriado.
9.7 Especificar a codificação dos caracteres para todas as conexões.
9.8 Filtrar os parâmetros que contenham informações sensíveis, provenientes do “HTTP
referer”, nos links para sites externos.

6.10 Requisito 10: Configuração do sistema

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

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

Subcontroles de Segurança Críticos - Configuração do sistema
10.1 Garantir que os servidores, frameworks e componentes do sistema estão executando a última versão aprovada.
10.2 Garantir que os servidores, frameworks e componentes do sistema possuem as atualizações mais recentes e seguras aplicadas para a versão em uso.
10.3 Desativar a listagem de diretórios.
10.4 Restringir, para o mínimo possível, os privilégios do servidor Web, dos processos e das contas de serviços.
10.5 Quando ocorrerem exceções no sistema, garantir que as falhas ocorram de modo seguro.

Subcontroles de Segurança Críticos - Configuração do sistema
10.6 Remover todas as funcionalidades e arquivos desnecessários.
10.7 Remover código de teste ou qualquer funcionalidade desnecessária para o ambiente de produção, antes de instalar o sistema no servidor de produção.
10.8 Prevenir a divulgação da estrutura de diretórios, impedindo que robôs de busca façam indexação de arquivos sensíveis, através da configuração do arquivo "robots.txt". Os diretórios que não devem ser acessados por estes indexadores devem ser colocados em um diretório isolado. Assim, apenas é necessário negar o acesso ao diretório pai definido no arquivo “robots.txt”, evitando ter que negar o acesso a cada diretório individualmente.
10.9 Definir quais métodos HTTP, GET ou POST a aplicação irá suportar, e se serão tratados de modo diferenciado nas diversas páginas da aplicação.
10.10 Desativar as extensões HTTP desnecessárias como, por exemplo, o WebDAV. Caso seja necessário o uso de alguma extensão HTTP com o propósito de suportar a manipulação de arquivos, utilize um mecanismo de autenticação conhecido, padronizado e bem testado.
10.11 Se o servidor processa tanto requisições HTTP 1.0 como HTTP 1.1, certificar-se de que ambos são configurados de modo semelhante ou assegure que qualquer diferença existente seja compreendida - como, por exemplo, o manuseio de métodos HTTP estendidos.
10.12 Remover informações desnecessárias presentes nos cabeçalhos de resposta HTTP e que podem estar relacionadas com o sistema operacional, versão do servidor web e frameworks de aplicação.
10.13 O armazenamento da configuração de segurança para a aplicação deve ser capaz de ser produzida de forma legível para dar suporte à auditoria.
10.14 Implementar um sistema de gestão de ativos para manter o registro dos componentes e programas.
10.15 Isolar o ambiente de desenvolvimento da rede de produção e conceder acesso somente para grupos de desenvolvimento e testes. É comum os ambientes de desenvolvimento serem configurados de modo menos seguro do que os ambientes de produção. Deste modo, os autores de ataques podem usar essa diferença para descobrir vulnerabilidades comuns ou encontrar formas de exploração.
10.16 Implementar um sistema de controle de mudanças para gerenciar e registrar as alterações no código, tanto de desenvolvimento, como dos sistemas em produção.
10.17 Rejeitar requisições que utilizem métodos (ou verbos) HTTP além do GET ou POST que não são utilizados pela aplicação (Verb tempering attacks), como, por exemplo, os verbos TRACE e OPTIONS e demais métodos.

6.11 Requisito 11: Segurança em Banco de Dados

A coleta e o tratamento de dados são tarefas fundamentais para que as organizações consigam, através de análises específicas, traçar metas e alcançar o objetivo final de atendimento aos usuários.
Desta forma, é fundamental que as organizações consigam garantir a proteção dos dados armazenados em banco de dados contra acessos indevidos, ações de hackers e incidentes técnicos.

Subcontroles de Segurança Críticos - Segurança em Banco de Dados
11.1 Usar consultas parametrizadas fortemente tipadas.
11.2 Utilizar validação de entrada e codificação de saída e assegurar a abordagem de meta caracteres. Se houver falha, o comando não deverá ser executado no banco de dados.
11.3 Certificar-se de que as variáveis são fortemente tipadas.

Subcontroles de Segurança Críticos - Segurança em Banco de Dados
11.4 Realizar a codificação (escaping) de meta caracteres em instruções SQL.
11.5 A aplicação deve usar o menor nível possível de privilégios ao acessar o banco de dados.
11.6 Usar credenciais seguras para acessar o banco de dados.
11.7 Não incluir strings de conexão na aplicação. As strings de conexão devem estar em um arquivo de configuração separado, armazenado em um sistema confiável e as informações devem ser criptografadas.
11.8 Usar procedimentos armazenados (stored procedures) para abstrair o acesso aos dados e permitir a remoção de permissões das tabelas no banco de dados.
11.9 Encerrar a conexão assim que possível.
11.10 Remover ou modificar senhas-padrão de contas administrativas. Utilizar senhas robustas (pouco comuns ou difíceis de deduzir) ou implementar autenticação de múltiplos fatores. Desativar qualquer funcionalidade desnecessária no banco de dados, como “stored procedures” ou serviços não utilizados. Instalar o conjunto mínimo de componentes ou de opções necessárias (redução da superfície de ataque).
11.11 Eliminar o conteúdo desnecessário incluído por padrão pelo fornecedor como esquemas e bancos de dados de exemplo.
11.12 Desativar todas as contas criadas por padrão e que não sejam necessárias para suportar os requisitos de negócio.
11.13 A aplicação deve conectar-se ao banco de dados com diferentes credenciais de segurança para cada tipo de necessidade - como, por exemplo, usuário, somente leitura, convidado ou administrador.
11.14 O Banco de Dados deve ser hospedado em um servidor diferente (virtualizados ou não) dos demais serviços.
11.15 Instalar, tão logo quanto possível, patches e hotfixes de versões mais atuais validadas e homologadas.
11.16 Encriptar os Backups de modo a proteger o sigilo das informações em caso de perda ou extravio.

6.12 Requisito 12: Gerenciamento de Arquivos

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

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

Subcontroles de Segurança Críticos - Gerenciamento de Arquivos
12.1 Não repassar dados fornecidos pelos usuários diretamente a uma função de inclusão dinâmica.
12.2 Solicitar autenticação antes de permitir que seja feito o carregamento de arquivos.
12.3 Limitar os tipos de arquivos que podem ser enviados para aceitar somente os necessários ao propósito do negócio.
12.4 Validar se os arquivos enviados são do tipo esperado, através da validação dos cabeçalhos, pois realizar a verificação apenas pela extensão é insuficiente.
12.5 Não salvar arquivos no mesmo diretório de contexto da aplicação Web. Os arquivos devem ser armazenados no servidor de conteúdos ou no banco de dados.
12.6 Prevenir ou restringir o carregamento de qualquer arquivo que possa ser interpretado ou executado pelo servidor Web.

Subcontroles de Segurança Críticos - Gerenciamento de Arquivos
12.7 Desativar privilégios de execução nos diretórios de armazenamento de arquivos.
12.8 Implantar o carregamento seguro nos ambientes UNIX por meio da montagem do diretório de destino como uma unidade lógica, usando o caminho associado ou o ambiente de "chroot".
12.9 Ao referenciar arquivos, usar uma lista de permissões (whitelist) de nomes e de tipos de arquivos permitidos. Realizar a validação do valor do parâmetro passado e, caso ele não corresponda ao que é esperado, rejeitar a entrada ou utilizar um valor padrão.
12.10 Não transmitir, sem nenhum tipo de tratamento, os dados informados pelo usuário a redirecionamentos dinâmicos. Se isso for necessário, o redirecionamento deverá aceitar apenas URLs relativas e validadas.
12.11 Não passar caminhos de diretórios ou de arquivos em requisições. Usar algum mecanismo de mapeamento desses recursos para índices definidos em uma lista pré-definida de caminhos.
12.12 Nunca enviar o caminho absoluto do arquivo para o lado cliente de uma aplicação ou para o usuário.
12.13 Certificar-se de que os arquivos da aplicação e os recursos estão definidos somente com o atributo de leitura.
12.14 Verificar os arquivos que os usuários submeterem através do mecanismo de carregamento em busca de vírus e malwares.
12.15 Limitar o tamanho máximo aceito para uploads de arquivos no servidor.

6.13 Requisito 13: Gerenciamento de memória

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

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

Subcontroles de Segurança Críticos - Gerenciamento de memória
13.1 Utilizar controle de entrada/saída para os dados que não sejam confiáveis.
13.2 Verificar se o buffer é tão grande quanto o especificado.
13.3 Ao usar funções que aceitem determinado número de bytes para realizar cópias, como strncpy(), estar ciente de que, se o tamanho do buffer de destino for igual ao tamanho do buffer de origem, ele não pode encerrar a sequência de caracteres com valor nulo (null).
13.4 Verificar os limites do buffer caso as chamadas à função sejam realizadas em ciclos e verificar se não há nenhum risco de ocorrer gravação de dados além do espaço reservado.
13.5 Truncar todas as strings de entrada para um tamanho razoável antes de passá-las para as funções de cópia e concatenação.
13.6 Na liberação de recursos alocados para objetos de conexão, identificadores de arquivo etc., não contar com o "garbage collector" e realizar a tarefa explicitamente.
13.7 Usar pilhas não executáveis, quando disponíveis.

Subcontroles de Segurança Críticos - Gerenciamento de memória
13.8 Evitar o uso de funções reconhecidamente vulneráveis, como printf(), strcat(), strcpy() etc.
13.9 Liberar a memória alocada de modo apropriado após concluir a sub-rotina (função/método) e em todos os pontos de saída.

6.14 Requisito 14: Práticas Gerais de Codificação

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

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

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

Subcontroles de Segurança Críticos - Práticas Gerais de Codificação
14.1 Para tarefas comuns, utilizar sempre código testado, gerenciado e aprovado, ao invés de criar código novo e não gerenciado.
14.2 Utilizar APIs que executem tarefas específicas para realizar operações do sistema operacional. Não permitir que a aplicação execute comandos diretamente no sistema operacional, especialmente através da utilização de "shells" de comando iniciadas pela aplicação.
14.3 Utilizar mecanismos de verificação de integridade por "checksum" ou "hash" para verificar a integridade do código interpretado, bibliotecas, arquivos executáveis e arquivos de configuração.
14.4 Utilizar mecanismos de bloqueio para evitar requisições simultâneas para a aplicação ou utilizar um mecanismo de sincronização para evitar condições de concorrência (race conditions).
14.5 Proteger as variáveis compartilhadas e os recursos contra acessos concorrentes inapropriados.
14.6 Instanciar explicitamente todas as variáveis e dados persistentes durante a declaração, ou antes da primeira utilização.
14.7 Quando a aplicação tiver que ser executada com privilégios elevados, aumentar os privilégios o mais tarde possível e revogá-los logo que seja possível.
14.8 Evitar erros de cálculo decorrentes da falta de entendimento da representação interna da linguagem de programação usada e de como é realizada a interação com os aspectos de cálculo numérico.
14.9 Prestar bastante atenção nas discrepâncias de tamanho de byte, precisão, distinções de sinal (signed/unsigned), truncamento, conversão e "casting" entre os tipos, cálculos que devolvam erros do tipo "not-a-number" e, também, como a linguagem de
programação trata a representação interna de números muito grandes ou muito pequenos.

Subcontroles de Segurança Críticos - Práticas Gerais de Codificação
14.10 Não transferir diretamente dados fornecidos pelo usuário para qualquer função de execução dinâmica sem antes realizar o tratamento dos dados de modo adequado.
14.11 Restringir a geração e a alteração de código por parte dos usuários.
14.12 Revisar todas as aplicações secundárias, códigos e bibliotecas de terceiros para determinar a necessidade do negócio e validar as funcionalidades de segurança, uma vez que estas podem introduzir novas vulnerabilidades.
14.13 Implementar atualizações de modo seguro. Se a aplicação precisar realizar atualizações automáticas, utilizar mecanismos de assinatura digital para garantir a integridade do código e garantir que os clientes façam a verificação da assinatura após descarregarem as atualizações. Usar canais criptografados para transferir o código a partir do host do servidor.

7. Exceções

Exceções às regras deste Procedimento serão tratadas pela Gerência de Infraestrutura e Operações juntamente com os Gestores apropriados a cada caso.

8. Utilização de Dados Pessoais (LGPD)

N/A

9. Anexos

Não há anexos